4-20-2021

Modeling Tools for Analyzing Electrical Power Distribution Systems Impacted by Electric Vehicle Load Growth

Jacob Sheeran
Portland State University

Recommended Citation

Sheeran, Jacob, "Modeling Tools for Analyzing Electrical Power Distribution Systems Impacted by Electric Vehicle Load Growth" (2021). *Dissertations and Theses.* Paper 5682.
https://doi.org/10.15760/etd.7554

Modeling Tools for Analyzing Electrical Power Distribution Systems Impacted by Electric Vehicle Load Growth

by

Jacob Sheeran

A thesis submitted in partial fulfillment of the
requirements for the degree of

Master of Science
in
Electrical and Computer Engineering

Thesis Committee:
Robert Bass, Chair
Richard Campbell
Jonathan Bird

Portland State University
2021

Abstract

Growing Electric Vehicle penetration presents unwanted problems to grid reliability. For nearly every EV there is corresponding household charging. High penetration of Electric Vehicle Service Equipment can lead to over loading of assets and under voltage conditions. In order to understand the effects EV have on a distribution system, studies have to be done for EVSE to understand how the distribution system is affected. Using advanced Power Engineering Simulation Software is often the best way to model systems due to the credibility of their software modules.

For this thesis, I developed a suite of distribution system analysis tools using CYME 7.1 and Python 2.7 for evaluating the impacts from EV penetration, particularly overloading and under-voltage events. EV penetration is the percentage of electric vehicles among total vehicles. Two of these tools apply new loads to and create intentional spotloads on the provided system. Another two tools incorporate time series demands for EV loads and provide load growth on the system. The final tool covers data collection for over loading and under voltage events.

Through use of this work's EV Evaluation Tools, users can study how a distribution system may be impacted due to EV load growth and stochastic EV placement. These tools allow for a representation of how the system changes with increased EV penetration, at the years the penetration are projected to increase.

Dedication

To the endless pursuit of knowledge.

Acknowledgements

Special thanks to Bob Bass, who showed me his passion through his teaching, for always pointing out where I need to improve and allowing me to grow, and for offering me the position to do complete this work. Thanks to Midrar and Shahad in the power lab who allowed me to bounce ideas off of them, and for Tyler, who helped me through some of my stranger problems when I was starting off.

Thank you to Andy Eiden, Ricardo Garcia and PGE for supporting my research, and helping me understand CYME.

I'd like to thank my mom and dad for always supporting me, and my girlfriend for putting up with my hours.

A big thanks to Johnathan Bird and Richard Campbell for being a part of my thesis committee and identifying areas that needed work. I really appreciate taking the time out their days to help me succeed.

Contents

List of Tables

List of Figures

Acronyms

AC	Alternating current. 13, 15
APF	Active Power Filtration. 22
CC	Constant Current. 14
CEC	California Energy Commission. 11
CV	Constant Voltage. 14
DC	Direct current. 13, 15
DCFC	Direct Current Fast Charger. 3, 13–15, 17
DOE	Department of Energy. 11
EIA	Energy Information Administration. 26, 27, 35, 37
EPRI	Electric Power Research Institute. 10
EV	Electric Vehicle. i, 4–9, 12, 13, 16–18, 23, 25–27, 29–31, 34, 35, 37, 39, 40, 42–46
EVSE	Electric Vehicle Service Equipment. i, 2–4, 6, 9, 12, 13, 15–17, 19, 22–32, 34, 35, 39, 40, 42–45
GHG	Greenhouse Gas. 1
GLOW	GridLAB-D Open Workspace. 11
Hi-P	High Power. 3, 4, 23, 28, 39, 40

HiPAS	High-Performance Agent-based Simulation. 11
ICE	Internal Combustion Engine. 1, 8, 23, 24, 35, 37
NEC	National Electrical Code. 6
NFPA	National Fire Protection Association. 6
OpenFIDO	Open Framework for Integrated Data Operation. 11
PES	Power Engineering Simulation Software. i, 10, 11
PF	Power Factor. 15
PGE	Portland General Electric. 1, 6, 7, 28
PNNL	Pacific Northwest National Laboratory. 10, 11
RECs	Residential Energy Consumption Survey. 27
SAE	Society of Automotive Engineers. 6
SOC	State of Charge. 14, 15
THD	Total Harmonic Distortion. 20, 21

1 Introduction

1.1 Problem Statement

In order to reduce and reverse climate change our emission sources need reduction. Climate change is by part due to emissions from Internal Combustion Engine (ICE) vehicles. In order to lessen the effects on the environment and create a healthier future, various states and countries have created electrification goals. PGE is driving and preparing for transportation electrification. Their reasoning is publicly presented by PGE [1]. The main goal of electrification plans are to reduce Greenhouse Gas (GHG) for the sake of improving the environment, lessening the impact of climate change.

Light duty vehicles represented 16.5% of total US GHG emissions, and 59% of transportation related GHG emissions in the USA in 2018 as seen in Figure 1.1 [2]. As consumer vehicles represent a large chunk of GHG production, many plans attempt to electrify light duty vehicles. Medium and heavy duty vehicles make up another 6.4% of total emissions as seen on Figure 1.1 making them another important source of electrification.

These electrification goals come during a period of increasing renewable generation, which further impact the electrical distribution systems. With the addition of stochastic renewable generation, line loading become less predictable. This can mean a larger impact on distribution lines if worst case scenarios with renewables are not considered and the

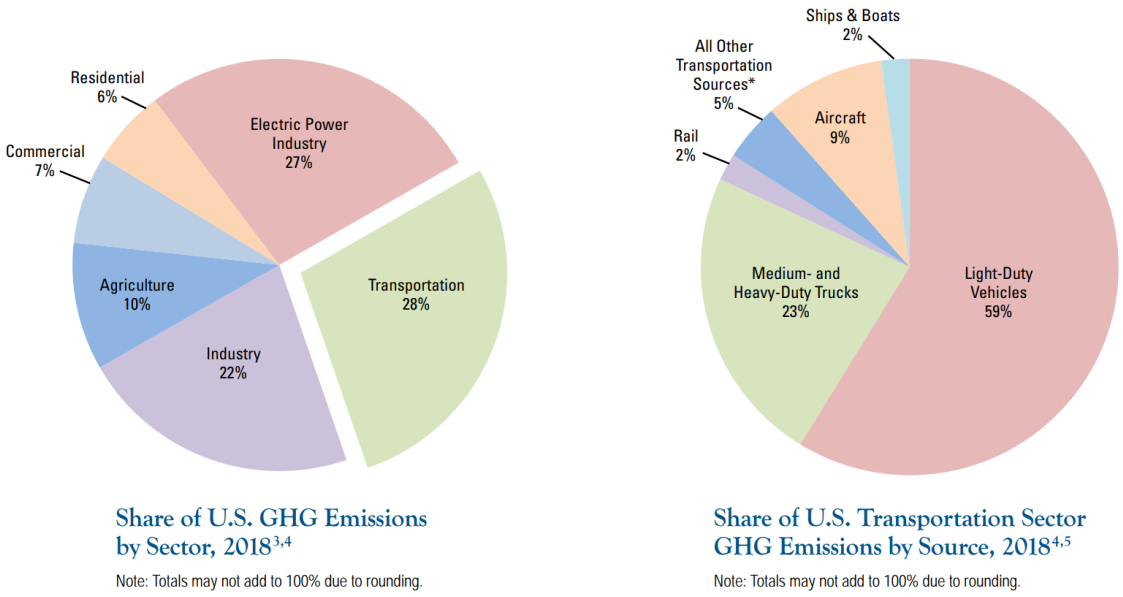

Figure 1.1: 2018 US GHG emissions by sector [2].

line rating doesn't reflect the real peak load. To maintain reliability, lines may need to be upgraded earlier than otherwise projected, in order to have sufficient line capacity to accommodate renewables and increased loading from Electric Vehicle Service Equipment (EVSE). In order to incorporate EVSE without overloading lines, these trends must be studied, especially in the case of high demand EVSE installations.

1.1.1 Details about Problem Statement

As the composition of passenger vehicles moves from gas to electric motors, new challenges have been identified. Household energy use may increase by up to 50% when they switch from an ICE to an electric vehicle [3]. These additions of EVSE to households bring not only a sizable increase in load, but also they may increase harmonics due to an inverters nature as a non-linear component. This work attempts to identify these problem areas and notify distribution engineers of the devices most at risk. As electrification continues, the

need for a reliable network of fast charging stations increases drastically [4]. Therefore, it is important to understand the impacts of these High Power (Hi-P) Direct Current Fast Charger (DCFC) installation on the current and future projected grids.

Harmonics and their impacts due to EVSE are pertinent to study, but are outside of the scope this thesis. Future work may be done to build on this work that incorporates harmonic analysis for a more comprehensive understanding of impacts on distribution systems. The literature review includes a section on harmonics for informative purposes, for readers and in order to inform future work.

1.2 Objectives of Work

Understanding the impacts that Hi-P EVSE have on distribution systems is the principle motivation that drove the development of these analysis tools. As vehicle ownership moves towards electric, a distribution system can experience problems if not prepared. Commuter vehicles may be one section of EVs, but have small batteries and charge relatively slowly, so the power draw is relatively low. With the introduction of electric taxi companies and electric semi-trailer trucks, time spent charging is loss of revenue [5]. For the sake of businesses, vehicles must be charged as quickly as possible to minimize downtime, hence there is a preference for Hi-P EVSE. These EVSE range from 50 kW up too 450 kW, a bound that will increase with time. It is important to understand how these Hi-P EVSE will effect the system prior to installing them to be better informed of what may happen. This ensures transformers are properly sized, or identifies if asset upgrades would be needed to safely

and reliably deliver the needed power.

Increases in residential Electric Vehicle (EV) penetration can also cause a variety of problems. As traditional loads grow year to year due increased electrification, it is fairly routine to project system impacts. However, EV loads act as an accelerator to load growth. It is important to keep EV penetration and load growth connected to provide the most accurate representation of the system loading. This effects the system at large. Stochastic allotment of EVSE and starting time creates a reasonable load shape for impact studies.

The tools developed for this thesis address two kinds of EVSE loads. One is the *Stochastic Residential EV Tool*, which applies residential EVSE additions to represent unplanned EVSE load growth. Second is the *Intentional EVSE Tool*, which creates EVSE charging hubs, for analyzing the how Hi-P EVSE additions affect the system, and the feasibility of intentional locations. These tools allow distribution planners to estimate how large an installation could be before an upgrade would be necessary, and allows them to plan for system expansion. These together allow a simulation of future grid impacts, to the extent the analyst can supply the EV growth predictions and load profiles.

Of the remaining tools, the *Time Series Tool* allows for time-series loads, pulling the projected EV profile from a supplied data file. This allows determination of length of overloads and daily occurrence. The *System Growth Tool* is concerned with finding the right percentage of vehicles to add onto a study, and non-EV load growth. This ensures a reasonable amount of vehicles are added for each EV penetration, and the distribution changes to reflect the new years forecast for EV Penetrations. Another tool, the *Data*

Collection Tool processes the loading data. The tool gathers the number and duration of overload events, then sorts assets by total number of overloaded events to provide a consistent format for distribution planners for each EV penetration.

1.3 Simulation Software Requirements

These tools were built using CYMDist version 7.1 as the main driver of simulations. Python 2.7 was used to create the code for these tools, and is recommended for use. Future versions of Python may require changing the scripts. In addition to CYMDist there is a module requirement. For interfacing with Python, *CYME Scripting Tool with Python* is a integral module. This module allows Python and CYMDist to interface, and these thesis tools to work. The *CYME Enhanced Substation Modeling* module can assist in more realistic simulations and is recommended.

2 Literature Review

The tools created for this thesis are related to many topics including EVs, how they impact equipment, and the software for simulation. EV standards used by PGE are referenced, as they affect how loading is considered within their balancing area. Different levels of EVSE, and their effect on the distribution systems reliability are recorded. The feasibility of converting fleets of vehicles to electric and the resulting impacts discussed. Information on different simulation programs is used to consider their positives and negatives. Harmonic effects are discussed in this review due to the importance of considering their impact. However, analysis tools that consider harmonics were not developed.

2.1 EVSE Codes & Standards

Various regulatory bodies have created standards that guide EVSE design and cope with its impacts on the electrical grid. These include the Society of Automotive Engineers (SAE) and the National Fire Protection Association (NFPA), which created the National Electrical Code (NEC). Utilities also create their own internal standards, such as for defining equipment overloads.

2.1.1 EV Service Equipment Charging Stations

PGE standard LD19300, *Charging Stations for Electric Vehicles*, provides design guidelines for EV charging stations [6]. A diversity factor as seen in Table 2.1 is applied while determining transformer power rating and conductor sizing. These factors are applied depending on the number of EVSE at each location.

Number of EV Charging Dispensers	Diversity Factor (%)
1 to 4	100
5 to 8	90
9 to 14	80
15 to 30	70
31 to 40	60
41 or more	50

Table 2.1: Diversity Factor Applied to EVSE Demand [6].

2.1.2 Transformers

LD17015, *Transformers General Physical and Electrical Characteristics*, is the PGE standard for transformer ratings [7]. It includes tables for each type of transformer, single and three phase, overhead or pad mounted. This standard includes the transformer ratings along with the size and space requirements for placing said transformer. This standard also provides a list of current limits depending on primary and secondary voltage.

2.2 Forecasting EV Use

Electric vehicle load growth, similar to other emerging technologies, is a matter of informed conjecture. The quality of prediction depends entirely on the validity of assumptions.

Forecasting techniques may be classified into two methods, bottom-up and top-down. These methods, while their results may be very similar, have very different approaches.

2.2.1 Top-Down Forecasting

Top-Down forecasting starts with a big picture, using information on similar devices to model how a device of interest will work, and works down to the desired information. As this method is based off large amounts of existing data making up the big picture, and EVs are a relatively new addition, forecasting methods have become more complicated in order to change recorded Internal Combustion Engine (ICE) data into a form that describes EVs. Forecast models are applied to the data in order to estimate what the future will look like. In many cases related to EVs, the data come from the National Household Travel Survey [8], an extensive collection of vehicle travel in the United States. This is a collection of information of when vehicles are moving, for how long, and over what distance. From these data, one can derive how an EV may behave, like charging time and state of charge. It is possible to relate these travel data into separate vehicle charging profiles, and stochastically apply them as done by Chioke et al. [9]. This effectively builds a model based off the ICE that incorporates EV characteristics.

2.2.2 Bottom-Up Forecasting

Bottom-up forecasting is an attempt to build a model based on the fundamental elements of the device, attempting to project the near future from current information. An example would be using the understanding of an air conditioning system to predict energy use due

to forecasted daily or hourly temperatures. Bottom-up forcasting has the disadvantage of requiring forecast data to inform the model. This method involves taking near-term forecast data such as daily temperature, creating individual loads, and then summing up the results. Bottom-up forecasting creates a well fitted model for each household, but is less robust and more variable then Top down forecasting [10]. Overall Bottom-up forecasting is limited due to a relative lack of EV studies while Top-down forecasting benefits from widely available traditional travel data.

2.2.3 Forecasting Usefulness

Forecasting models, regardless of method, provide important insight into how distribution systems may transition over time. One of the most valuable takeaways would be EV load shapes for different classifications of vehicle, load shape being the shape of the demand profile of EVSE throughout the day. These could then be aggregated to provide an estimate of the load throughout a distribution system during specific time periods. Figure 2.1 below shows the forecasted load shapes for South Korea, showing a residential shape, and a commercial shape for non-residential chargers [3]. One of the most important benefits of forecasting methods is the estimation of load demand. This allows an estimation of system load, which is useful for generation dispatch planning. Both Top-down and Bottom-up methods can lead to similar results, depending on the size of the study the forecasts applied. For small numbers of forecasted devices, Top Down often leads to better predictions due to having less variance than the Bottom-up method [10]. For larger systems, the error due to either method is fairly low, due to the law of large numbers evening out variance.

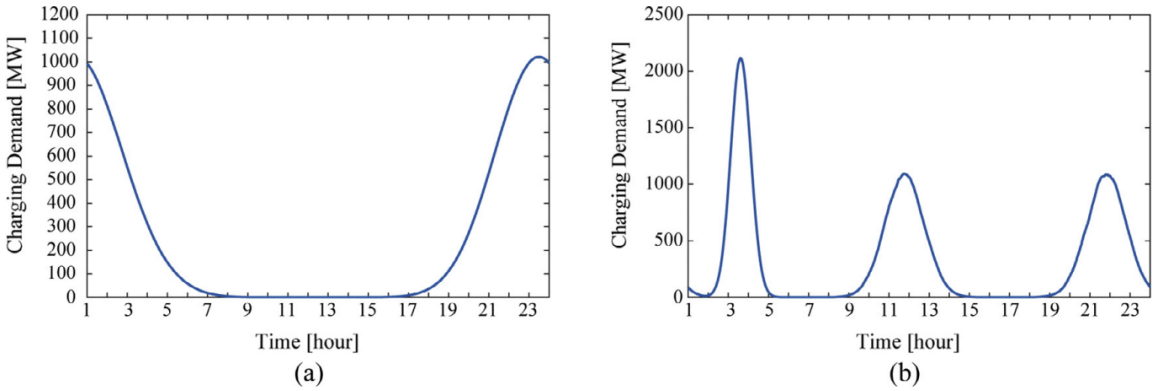

Figure 2.1: Weekend EV load shape projections in South Korea (a) Residential (b) Commercial [3].

2.3 Power Engineering Simulation Software

PES (Power Engineering Simulation Software) allows for the simulation of a distribution system. Some specific examples of commercial PES software companies include CYME and ETAP. Non-commercial tools, such as GridLAB-D created by the Pacific Northwest National Laboratory (PNNL), and OpenDSS, by the Electric Power Research Institute (EPRI). Examples of PES functionality include analyzing transformer inrush, and compliance studies for arc flash standards. Without an established PES, users would have to create their own software for modeling every relevant component on the system, taking a long time with little benefit. Various vendors of PES often offer the same base functionality, but offer different modules depending on interests. In this section, I present CYMDist as well as another tool that I considered, GridLAB-D.

2.3.1 CYMDist

CYME is the power engineering software company that created CYMDist, an advanced analysis module for distribution power systems. Their products are currently used by utilities

as well as power systems researchers [11] [12]. CYME offers code-compliant simulation software that incorporate many methods of power system analysis. CYME provides means for applying adopted techniques through modifications of CYME modules [13].

CYMDist was used as the framework of my analysis tools for a couple of reasons. First of all it is popular, which lends towards its user accessibility, as well as its capabilities. Due to its popularity, CYME provides a large user base for these tools. The major downside of CYME is that it is an expensive software package, which prevents widespread use among researchers and small utilities.

2.3.2 GridLAB-D

GridLAB-D is a power system simulation tool created by PNNL with funding from the Department of Energy (DOE). This tool was released to the public in 2008 at a time when smart grids were an emerging idea and had not become as developed as they are now. GridLAB-D was developed to help break the barrier to adoption of smart grid techniques. Since its creation, GridLAB-D has been updated regularly, including an overhaul in 2017 funded by the California Energy Commission (CEC). The overhaul focused on GridLAB-D Open Workspace (GLOW), a user interface, High-Performance Agent-based Simulation (HiPAS) to enhance GridLAB-D performance, and the Open Framework for Integrated Data Operation (OpenFIDO) for supporting data exchange with other power analysis software. There is current work on it, such as by Nasiakou et al., which proposes a GUI for GridLAB-D, improving accessibility [14]. GridLAB-D is one of the most widely available PES and improvements are still occurring, attempting to create a more accessible framework [15].

GridLAB-D, while being free was not chosen for a number of reasons. GridLAB-D has a less extensive reference library then CYME, and less information online in general due to their scales. GridLAB-D lacks an established Python library, which would assist in tool development, though GridLAB-D can interface with Python scripts using the HELICS environment as an intermediary.

2.4 Electric Vehicle Service Equipment

EVSE is the term for the connection point between the vehicle and the power source. EVSE are classified into three levels. These levels represent different power levels, classified as Level 1, Level 2, and Level 3. Level 1 and 2 EVSE work in conjunction with an EVs internal charging equipment, which performs AC to DC conversion inside the EV. Level 3 EVSE however, convert AC to DC outside of the EV before supplying DC voltage and current to the vehicle. The standards for AC EVSE can be seen in Table 2.2, DC EVSE standards can be found in Table 2.3.

Charge Method	Voltage (AC V)	Phase	Max. Current (A, continuous)	Branch Circuit Breaker Rating	Max. Power (kW)
AC Level 1	120	1-phase	12 16	15 (min.) 20	1.44 1.92
AC Level 2	208 to 240	1-phase	<60	Per NEX 625	Up to 19.2

Table 2.2: SAE J1772 AC EVSE Standards [16].

Charge Method	EVSE DC Output Voltage (DC V)	Max. Current (A)	Max. Power (kW)
DC Level 1	50 to 1000	80	80
DC Level 2	50 to 1000	400	400

Table 2.3: SAE J1772 DC EVSE Standards [16].

2.4.1 Level 1 and 2 Chargers

Level 1 chargers have the lowest power rating and therefore the lowest individual impact on a distribution system. They accept 120 V and have a maximum current of 12-16 A, allowing users to directly plug their EVSE into a standard receptacle. These Level 1 chargers, while versatile, and are mostly used for overnight charging due to slow charging speed.

Level 2 chargers provide higher power then Level 1 chargers. They require 240 V or 208 V and have much higher current ratings than Level 1, 32-80 A depending on the standard cited. Because of their higher power rating, these charge faster then a Level 1 charger.

2.4.2 DC Fast Chargers/Level 3

Level 3, also known as DCFC, operate on a different basis than Level 1 and 2, as it uses DC current. The conversion from AC grid power to the DC battery charging occurs within the service equipment. A simplified block diagram for DCFCs is shown in Figure 2.2, showing the rectification from the grid AC. DCFCs can vary considerably, from a standard 50 kW charger, to the 250 kW Tesla V3 supercharger, and upward to 1 MW for Tesla Semi-Truck Chargers. Due to their power, DCFCs are best for charging quickly, as they are able to provide a nearly full charge in a half hour, depending on the class of EV.

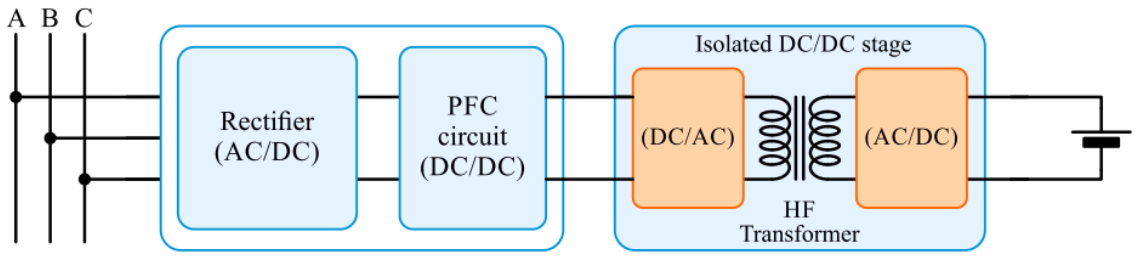

Figure 2.2: A simplified block diagram of a state-of-the-art dc fast charger power stage [17].

2.4.2.1 Demand Charges

Depending on the local utility, demand charges may be included in electrical bills [18]. A demand charge is a payment based on the highest single point of consumption within a billing cycle. As DCFCs typically consume 50 kW or more, this can raise a customers peak demand dramatically. This is exacerbated as DCFCs are placed together. Multiple DCFCs may be used concurrently, thereby resulting in a much larger single point of consumption. In areas with demand charges, utility bills can increase by a factor of four [18]. This is an important quality to note for consumers who are subject to demand charges, as it could be a financial burden. Due to financial burden understanding of demand chargers may be a barrier to transportation electrification.

2.4.3 EVSE Charging

Charging cycles are split into two periods, Constant Current-Variable Voltage (CC), and Variable Current-Constant Voltage (CV) [19]. The CC phase injects current into the battery in order to raise the battery voltage level. This period of charging occurs at maximum current without risk of damaging the battery during the low SOC period of the charging cycle. The CV phase starts once voltage increases to rated value. The current decreases in order to allow safe charging of the battery at a higher State of Charge (SOC). The current decrease allows charging without damaging the battery while at a high SOC. Due to damage concerns, batteries may only be charged to 80-90% of their maximum charge. This increases the lifespan on the battery. This also assists with charging rates, the last 20% taking around

the same amount of time as the first 80% SOC [20].

2.4.3.1 Power Factor Correction

EVSE rectify power from the grid AC to vehicle DC. Rectifiers are considered non-linear equipment, and as such cause distortion while converting AC to DC. Level 1 and Level 2 chargers often have simpler rectifiers that act as the PFC stage itself [21]. DCFCs as seen in Figure 2.3 separate the rectification and the PFC circuit. As observed by Pawelek, et al. the Power Factor (PF) of a DCFC is not consistent throughout the charge cycle, in their case reducing from 0.99 to roughly 0.80 [20]. This drop in PF only occurs after the start of the constant voltage section of battery charging, as imported power decreases along with current.

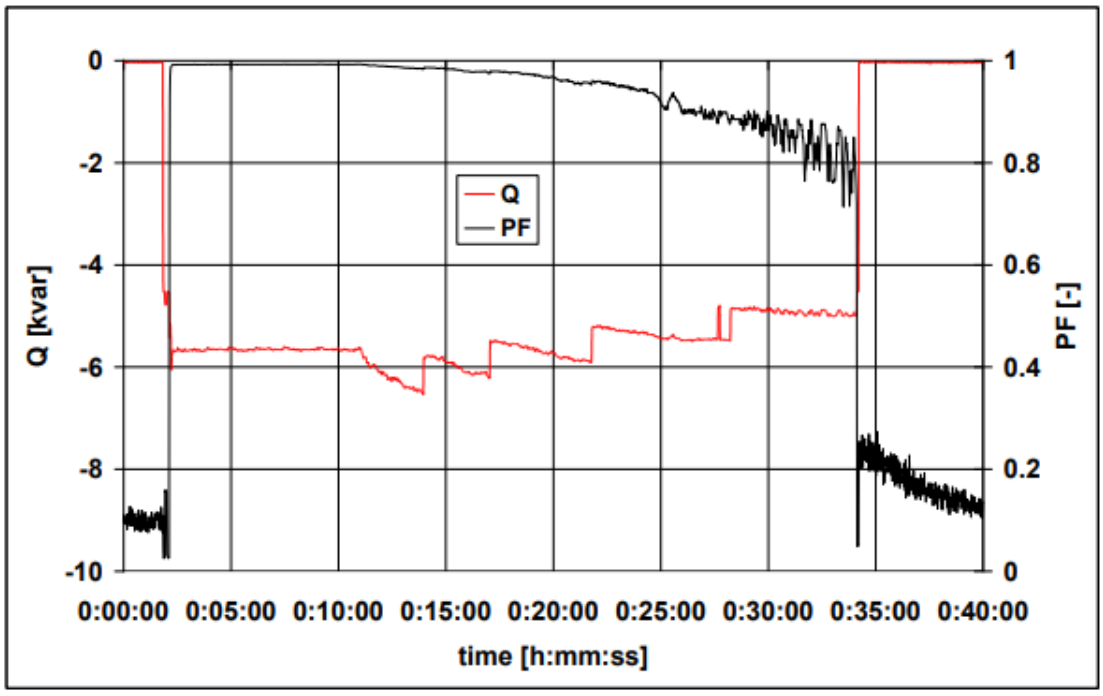

Figure 2.3: Power factor and reactive power for DCFC charging cycle. [20].

2.5 EVSE Impacts on Power System Reliability

Charging impacts, especially for residential EVSE, depend on if charging is coordinated or uncoordinated. Uncoordinated charging, letting the households charge their EV at any time is a simple method. Most customers are used to being able to do things when they wish, not held back by the utility. This method requires the utilities to do little until more significant distribution system impacts show themselves, and distribution upgrades are required. The tools created for this thesis examine the impacts of large scale EVSE to asset loading and voltage magnitudes with uncoordinated charging.

Coordinated charging allows operators to control EVSE use. Controlling charging would allow grid operators to shift load shape in order to ensure reliability. Clement-Nyns, et al. compare the impacts of uncoordinated and coordinated charging of hybrid vehicles along with base case with no EVSE and find clear effects on the grid [22]. As seen in Table 2.4 and Table 2.5, nodal voltage and power losses are both improved with coordinated charging. Coordinated charging offers grid impacts minimal compared to uncoordinated charging with respect to the base case.

Ratio of Power Losses to Total Power [%]					
Charging Period	**Penetration Level**	**0%**	**10%**	**20%**	**30%**
21h00-06h000	Summer	1.1	1.4	1.9	2.2
	Winter	1.4	1.6	2.1	2.4
18h00-21h00	Summer	1.5	2.4	3.8	5.0
	Winter	2.5	3.4	4.8	6.0
10h00-16h00	Summer	1.3	1.8	2.6	3.2
	Winter	1.7	2.2	3.0	3.6
Minimum Voltage Deviation [%]					
Charging Period	**Penetration Level**	**0%**	**10%**	**20%**	**30%**
21h00-06h000	Summer	3.1	3.5	4.4	5.0
	Winter	4.2	4.4	4.9	5.5
18h00-21h00	Summer	3.0	4.4	6.5	8.1
	Winter	4.8	6.3	8.5	10.3
10h00-16h00	Summer	3.0	4.1	5.6	6.9
	Winter	3.7	4.9	6.4	7.7

Table 2.4: Ratio of power loss and maximum voltage deviation for uncoordinated charging [22].

2.5.1 DCFC Impacts

EVSE increase EV charge time, but have their own impacts. One problem associated with EVSE is voltage deviation, wherein voltage decreases at near-by nodes. As shown by Lillebo et al., EVSEs can reduce the per unit voltage by up to 2% due to line voltage drop, bringing possible concerns with EVSE placement [23]. On a stiff feeder, this would not cause much of a problem, but on a feeder already facing voltage tolerance problems, this could cause under voltage breakers to trip. Another study by Clement-Nyns et al. looked at how EVSEs impact weaker buses [24]. Bus strength was determined as per unit voltage drop due to increased bus loading. A per unit tolerance value of ±5% may be used to classify under and over voltage events. The weakest bus had a per unit voltage drop of 0.17, well below acceptable levels. In the case of weaker busses, Level 2 chargers are recommended over DCFCs to avoid such voltage drops.

Ratio of Power Losses to Total Power [%]					
Charging Period	**Penetration Level**	**0%**	**10%**	**20%**	**30%**
21h00-06h000	Summer	1.1	1.3	1.7	1.9
	Winter	1.4	1.5	1.8	2.1
18h00-21h00	Summer	1.5	2.3	3.7	4.7
	Winter	2.4	3.3	4.7	5.8
10h00-16h00	Summer	1.3	1.7	2.3	2.8
	Winter	1.7	2.1	2.7	3.2
Minimum Voltage Deviation [%]					
Charging Period	**Penetration Level**	**0%**	**10%**	**20%**	**30%**
21h00-06h000	Summer	3.1	3.1	3.3	3.7
	Winter	4.2	4.2	4.2	4.3
18h00-21h00	Summer	3.0	4.1	5.8	7.2
	Winter	4.8	6.0	7.8	9.1
10h00-16h00	Summer	3.0	3.3	4.1	4.7
	Winter	3.7	4.0	4.9	5.5

Table 2.5: Ratio of power loss and maximum voltage deviation for coordinated charging [22].

2.6 Harmonic Effects

While this thesis does not specifically address harmonics, harmonic functionality may be developed in the future. Electric vehicle charging involves non-linear power electronics. These components present a non-linear relationship between current and voltage. Woodman et al. describe the harmonic impacts of equipment specifically due to EV charging [25]. In general, due to the increases in frequency at higher orders, and of current due to distortion, harmonics can cause heating, improper relay and fuse activation, and loss of equipment life. However, Woodman et al. found that EVSE impacts on distribution assets are generally small enough to neglect, in most cases.

2.6.1 Harmonic Cancellation

In the case of multiple EVSE, each EVSE may create their own harmonics which interact; what the grid experiences is not a sum of these harmonics. This is due to harmonic cancellation, that certain harmonic order contributions from each charger can cancel each other out due to phase differences. Studies of harmonic cancellation have found that harmonics in magnitude and phase angle are effected by each new EVSE connected for charging, and that the cancellation effect is time dependent. Malano et al. define primary cancellation as the phase angle diversity, reducing harmonic magnitudes due to difference in phase angle reducing magnitude [26]. Secondary cancellation is defined as the impact on harmonic voltage drop caused be total harmonic current on the harmonic voltage background, as the voltage drop can increase or decrease the respective voltage harmonic. Figure 2.4 presents the changes in current harmonics due to harmonic cancellation, Figure 2.5 shows the changes in voltage harmonics. It is important to note the 7th harmonic for both voltage and current, as they show trend reversals for harmonic changes as EVSE start charging. These figures demonstrate that voltage and current harmonics change in magnitude and phase angle due to multiple EVSE harmonic emissions.

2.6.2 Transformers

Transformers are susceptible to harmonic disturbances. High order harmonics can lead to high I^2R losses, which increase real power consumption and decrease power factor [27]. Harmonics can induce additional eddy current and hysteresis losses in the transformer

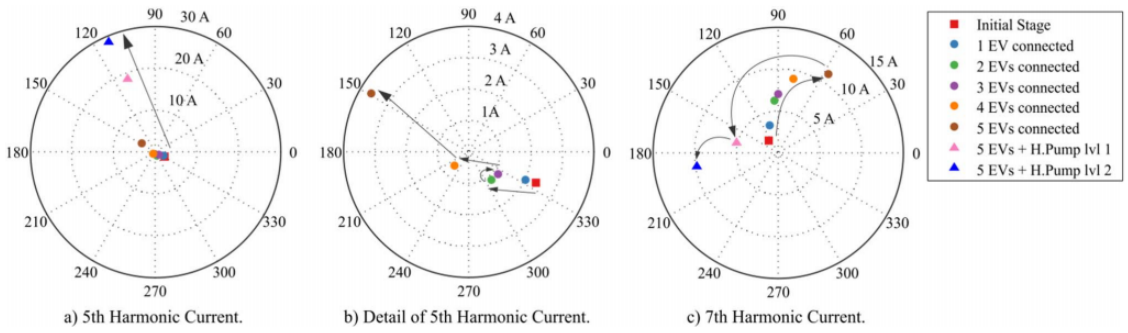

Figure 2.4: Current harmonics during coordinated charging [26].

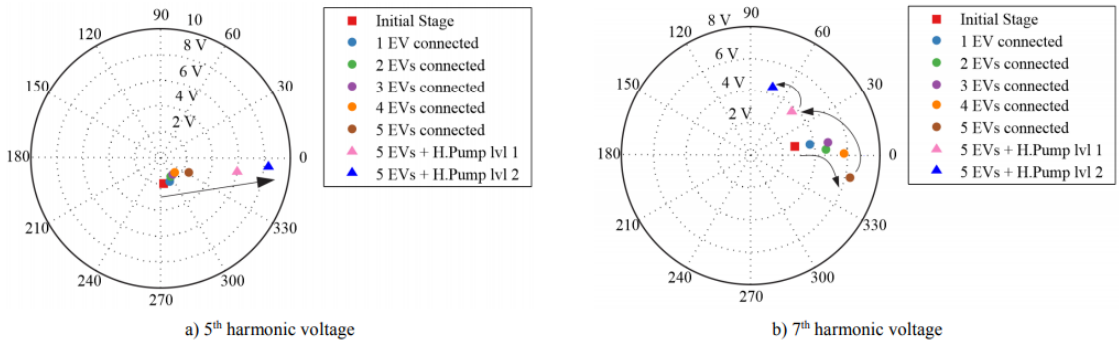

Figure 2.5: Voltage harmonics during coordinated charging [26].

core. Eddy current and hysteresis losses scale with f and f^2 respectively. This results in enhanced effects due to higher order harmonics, which are usually ignored due to their low magnitudes, relative to the fundamental and lower order harmonics. These two losses cause temperature increases inside the transformer, and accelerates insulation degradation, which in turn reduces transformer life span [28]. Along with the losses in the core and resistive losses, the magnetic components experience stray flux from harmonics, which induce larger eddy current and hysteresis losses, contributing further to equipment heating and loss of life. Gómez et al. examine the relationship between Total Harmonic Distortion (THD) and transformer per unit life consumption [29]. Per unit life consumption effectively represents excess transformer component damage. Figure 2.6 shows a plot of transformer

life consumption in respect to THD ranging from 5 to 100%. This figure includes a quadratic best fit line, showing the rapid increase of life consumption at high THD levels. The authors recommend keeping THD limited to 25-30% for a given transformer, in order to maintain an acceptable lifespan.

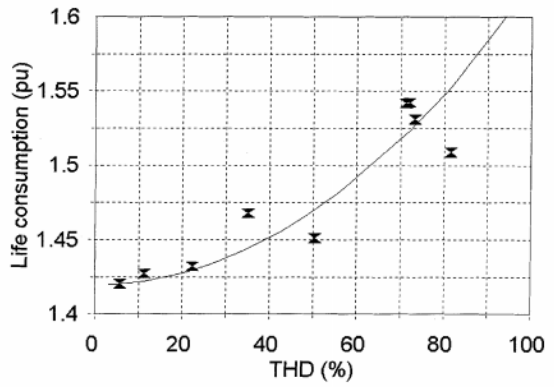

Figure 2.6: Transformer Life Consumption vs. THD [29].

2.6.3 Power Cables

Cables suffer two frequency related problems, namely the proximity effect, resulting in current crowding, and the skin effect, which forces current to the outside layer of the current carrying wire. These effectively raise the resistance of the conductor to match the reduced volume of the current-carrying metal. As both the effects are related to frequency, their impacts are increased for the higher order harmonics. In effect, the higher order the current the larger the increase in resistance. This increase translates to more real power losses, and increased line voltage drop.

2.6.4 Filters

As harmonic orders are organized by their frequency, filters can be a method of reducing harmonic effects. Syed Nasir et al. describes the effects of filters in reducing total harmonic distortion [30]. They examine optimally-placed passive filters. Analysis of the capabilities of passive filters classes are expressed in by Karadeniz et al. [31]. Placing a filter to remove identified problem harmonics can alleviate much of impact. Karadeniz et al. explores the relationship between harmonics and the de-rating of transformer rated power [31]. Low-pass filters are used generally to remove high order harmonics, with band-stop filters generally being used to remove specified harmonics. The most powerful form of filters, Active Power Filtration (APF) could be useful for larger EVSE installations, but is used less often than passive filters due to their power requirements and cost.

2.6.5 Manufacturer Value

Harmonic profiles are offered by some manufacturers of EVSE. Thiringer et al. measured harmonic values and compared them with the manufacturer supplied harmonics [32]. While most of the harmonics they tested showed little difference to the supplied values, the 5th harmonic was 71% larger than expected when measured. Manufacturer supplied values should be considered, but not relied upon when considering harmonic impacts of distribution systems.

2.7 Fleet EVs

As electrification increases and electric vehicles take the place of traditional ice vehicles, commuters are not the only ones making the switch. Company fleets will also make the change from ICE to EV. This of course has many benefits for the fleet owner: less expense, maintenance, lower pollution, on-site charging, and a greener business profile. Distribution systems face a different impact due to fleet EVs, however, with increased load due to a large number of new EVSE. In the case of using Hi-P EVSEs, large grid impact on distribution equipment loading and voltage levels may occur. These impacts need to be studied prior to a utility issuing an interconnection agreement.

2.7.1 Driving Services

Services like taxis could benefit from conversion to EV, with low average trip distance and city range constraints allowing ample access to charging infrastructure [33]. EVSE allow drivers to charge when they need to during the day, and while waiting for a customer. The only real disadvantage is the speed of charging, as conventional vehicles fuel up in a matter of minutes. A taxi company could use EVSEs to decrease charge time, if the have the capital to set up chargers specifically for their taxis. One taxi company recorded their previous travel data and found the average daily travel distance averaged less then 250 km, a fairly standard EV range [33]. They found due to the large idling time of up to 20 hours per day, there is value in including both fast and slow chargers at taxi hubs.

Similar to taxis, ride sharing apps observe many of the same benefits as company fleets

[34]. Cost per mile is the largest benefit, EVSEs being cheaper to operate per mile then ICE. This allows ride share users to make a better profit, pocketing the price savings or reducing costs to maintain competitiveness. In the case of self driving vehicles, fast charging through ride sharing programs will serve a similar purpose to the taxi charging hub. With full autonomous driving, fleets can take even further steps to optimize fleet activity such as in [35]. Another benefit comes from having quiet transportation, which is a positive quality that could make electric vehicles a preference over ICE vehicles on ride sharing apps.

3 Design Considerations

Design Considerations are guidelines that influence the design of tools. These arise from consideration of the desired outputs and how to achieve them. These considerations influence the design of each tool and how they work together. The following sections document the design considerations that were taken into account while designing this suite of tools.

Five tools were developed for this project. They include a *Stochastic Residential EV Tool*, which is needed to represent residential vehicles and their EVSE. The *Intentional EVSE Tool* is used to represent non-Stochastic EV growth. In order to allow distribution systems to simulate different years, the *System Growth Tool* was created. The *Time Series Tool* allows time series analysis and the use of demand profiles. EVSE impacts are gathered using the *Data Collection Tool*, which collects loading and voltage data.

3.1 Stochastic Residential EV Tool

The *Stochastic Residential EV tool* was created to gather the total number of vehicles on the distribution system and apply EVSE loads to each household. When given a distribution study, this tool produces demand increases via EVSE at each residential household stochastically. Given an input of compositions relating household demand to vehicles at a house, this tool outputs the number of vehicles for each household.

Two considerations stand out: one, how to add the loads being applied, and two, how many vehicles each household should have. These considerations suggest that this tool should add residential EVSE stochastically within a distribution system model, and estimate the number of vehicles that should be at each household. An estimation of the number of possible EVs is important in determining EV penetration through applied vehicles.

3.1.1 Stochastic Placement of Residential EVSE

Placement of EVSE is an integral part of these tools. In order to achieve this, each vehicle needs a given chance to increase the EVSE demand for the customer. Each customer must have the capability of adding multiple EVSE, to represent different EVSE for the customer. A customer with no initial demand should be created at each spotload to hold EVSE demand, to separate household demand from EVSE demand. The level of charger being applied is determined from the composition of Level 1 to Level 2 EVSE.

3.1.2 Household Vehicle Modeling

In order to simulate a distribution system, the number of vehicles on the system must be estimated. There are two quantities within CYME that could be related to the number of vehicles in a household: the *number of customers*, and the *electrical demand* in kW. As the *number of customers* indicates the number of billed accounts, not number of people living at that location, this value is not used. The *electrical demand* of each household can be read from a CYME model, which provides the average hourly demand. Census and Energy

Information Administration (EIA) data may give us an approximation of the number of vehicles at a household.

3.1.2.1 Modeling Household Members By Load

In order to relate number of adults per household to household demand, EIA 2015 Residential Energy Consumption Survey (RECs) micro-data were used. Total power consumption and number of adults at each household can be compared using EIA data [36]. These data can be used to achieve a rough composition of the different household member sizes by demand.

3.1.2.2 Modeling Household Vehicles By Members

Once the household sizes have been calculated, another relationship is required to connect household size to number of vehicles. This relationship allows each household to have an amount of vehicles that may become EVs. US Census data may be used for their estimation of household vehicle composition. The total number of vehicles on the distribution system can be used to reflect EV penetration.

3.2 Intentional EVSE Tool

The purpose of the *Intentional EVSE Tool* is to place specified EVSE on the distribution system based on user input. Intentionally placing EVSE onto the system with this tool allow non-stochastic EVSE placement. These intentional EVSE also need to have the ability to be delayed, taking a user input of EVSE installation date, to add the EVSE into the distribution

study during a specific year. Functionality for placing stochastic as well as non-stochastic EVSE will provide a more comprehensive view of EVSE impacts on the distribution system.

Many design considerations were considered: where to place Hi-P EVSE, how to attached the spotload to the distribution system,and how to represent future EVSE installations. The first two considerations allow Hi-P loads to be placed on the distribution system at a specific location, with a transformer designed for the load. The final consideration allows EVSE installations to be applied to the study at a specified year. Overall, this allows representation of EVSE installation on the distribution system while considering where and when it will be connected to the distribution system.

3.2.1 Hi-P EVSE Loads

Hi-P EVSE need to be placed at a specific location designated by the modeler. In order to achieve this, the name of an existing spotload is used. Spotloads are placed at specific electrical connection points within a model. Hi-P EVSE charging stations often have their own transformer for handling the large EVSE load. Each intentional EVSE installation needs its own transformer, so the *Intentional EVSE Tool* must to be able to create the assets connecting the spotload to the distribution system.

3.2.1.1 Diversity Factor

In order to incorporate PGE's standard for transformers, a diversity factor must be applied to the EVSE transformer installation. This factor depends on the number of similarly-sized chargers at the location. Diversity factor accounts for the assumption that the full connected

capacity will not be in use all at once. This factor will need to be applied to demand before placing the transformer on the distribution system, to create properly-sized transformers for the EVSE.

3.2.1.2 Planning

EVSE installations often take time to develop. Projects may be planned many years in advance before installation. In order to incorporate this time lag, there must be an option to set a delayed commission year while inputting load information. This enables adding the installation to the study at an appropriate time as dictated by the operator.

3.3 System Growth Tool

The purposes of the *System Growth Tool* are, given an EV penetration, to calculate the number of vehicles added to the system, and to represent with the the distribution system the expected EV penetration in every year of the study. The projected year is used as an input to increment non-EVSE loads across the system. This non-EVSE load growth is a result of continued electrification due to increased household electrical demand each year. Reflecting both EVSE and non-EVSE load growth on the distribution system creates a more accurate estimation of a future system.

The design considerations for this *System Growth Tool* include applying rational EVSE load growth estimates and applying non-EV load growth estimates to the distribution system. The first consideration allows EVSE growth reflecting the EV penetration by EVSE application. The second consideration enables applying non-EV load growth to successive

years of projected EV penetration. These considerations allow the system to reflect the changes across different years.

3.3.1 Residential EV Penetration Increases

To represent changes in EV penetration, this tool must be able to apply the correct number of vehicles. This value is based on the number of remaining vehicles that may to be applied to the study. As the number of vehicles in the study decreases, the percentage of vehicles applied must increase in order to represent the same EV penetration increase.

In the case of a distribution system with the EV penetration already known, different calculations are used for determining the percentage of vehicles applied to the system. There is a need to adjust the EVSE additions, to account for EVSE on the original study. This adjustment must used to prevent excess EVSE additions to a distribution study.

3.3.2 Non-EVSE Load Growth

In order to represent the distribution system load growth through time, non-EVSE demand must increase. Due to general electrification, average household demand increases each year. Where non-EVSE growth occurs each year, EVSE growth occurs whenever EV penetration changes. This non-EVSE load growth along with EVSE load growth provides a more representative prediction of impacts on the distribution system.

3.3.2.1 EVSE

EVSE load growth is driven by increases in EV ownership, while household load growth is driven by non-EVSE electrification trends. As EVSE and non-EVSE loads experience growth through two different phenomena, there is a need to exempt EVSE from non-EVSE load growth. Seperating EVSE load growth from non-EVSE load growth allows EVSE demand increases to be attributed to households the include EVSE. This also allows for different load growth rates.

3.4 Time Series Tool

To allow time series simulations and produce time-series records, the *Time Series Tool* was created. Given an input of demand profiles, this tool must be able to access the demand profiles across supplied time steps. These demand profiles are needed to find duration of overloaded events. Considerations were taken to allow: 1) time-series demand profiles, 2) time-series distribution system demand changes due to demand profiles, and 3) consistency of vehicle records. These considerations allow time series demand profiles to be applied to the distribution system, and their impact on assets to be recorded.

3.4.1 EVSE Charging Demand Profiles

In order to use demand profiles, the modeler must supply many different EV profiles. EV demand profiles are used to represent a time series load. These profiles allow projected EVSE charging demand profiles to be represented on the distribution system. The impacts

from these profiles show how the system may react during specified time steps for loading and voltage levels.

3.4.1.1 Intentional EVSE Charging Demand Profiles

As intentionally applied EVSE spotloads may have any composition of EVSE a generic profile can not represent the wide variety of EVSE. Operators must be able to create their own profiles specific to the EVSE. This allows a more accurate representation of how the distribution system is affected by intentionally added EVSE.

3.4.2 Time Series Changes

Two functions were created to deal with time series changes. The first function handles applying new EVSE demand changes. This function must be able to access EVSE demand profiles to find the demand at a given time step. A second function gathers and records the voltage and loading values per phase for each asset. These two functions allow demand profiles to be used and their impacts to be recorded.

3.4.3 Information Storage

Using demand profiles requires that each vehicle keep a consistent demand profile index. This is solved through keeping records for each household. If the records restrict which values can be changed, demand profile will stay consistent. These records may be used to store relevant information to vehicles, such as EVSE level and current type of vehicle ICE or EV.

3.5 Data Collection Tool

The purpose of the *Data Collection Tool* is to record asset values on the distribution system. Gathering this asset information allows EV impacts to be accessed. Design considerations for this are first, gathering the loading and voltage data for each asset, and second, making sure these recorded data are stored in an accessible format. This *Data Collection Tool* allows the loading and voltage values for each asset to be recorded, with records easy to access.

3.5.1 Asset Loading and Voltage Collection

This suite of tool looks for overloading and under-voltage events. In order to determine when these events occur and for how long, asset loading and voltage levels are recorded for each phase. Showing each of the phases gives a better picture for the operator into impacts on the individual phases, and possible unbalanced phases in each asset.

3.5.2 Stored Loading and Voltage Format

Each loading and voltage quantity needs to be placed into different records to allow overloading and under-voltage events to be analyzed individually. These records should use the same file format to allow non-asset specific functions. These loading and voltage records should be stored in order of occurrence to make accessing different EV penetrations or time steps simple. Each asset class may be stored in a separate record to allow ease of analysis.

4 Tool Development

The following subsections document how each design consideration was realized. Each section begins with an explanation of what the tool should be able to do. Subsections then present the specific design considerations. Each of these subsection explains how the tool achieved the consideration.

4.1 Stochastic Residential EV Tool

The *Stochastic Residential EV Tool* is used to apply EV loads and to set an estimate on the number of possible vehicles in a household. The following sections describe how EVSE loads are placed onto a distribution system, and how the number of possible vehicles for each household is determined. These capabilities ensure stochastic EVSE application as well as a reasonable number of vehicles for each household.

4.1.1 Stochastic Modeling of Residential EVSE

Stochastic EVSE modeling refers to placing EVSE randomly across the distribution system. Every spotload on the distribution system is checked to find residential customers. If a spot load represents a residential customer, a random number is compared to the percentage of EV to be placed on the distribution system. Spotloads have a chance of adding EVSE for

each ICE vehicle they are estimated to have. This is used to simulate converting from ICE to EV.

When adding an EVSE, the previously stored household records are referenced to determine the level of EVSE. Level 1 and Level 2 chargers use a rated power of 1.92 kW and 6.6 kW, respectively. Demand profiles are provided Level 1 and Level 2 chargers, other demand profiles can be input by the operator. When EV growth is applied, the first time step of EVSE demand profiles are applied as the demand. Figure 4.1 shows the flowchart for the decision paths made inside the EV application step.

4.1.2 Household Vehicle Modeling

In order for the tool to estimate the number of vehicles attached to each household, household electrical demand is related using US Census and EIA data [36]. The EIA provides energy consumption data over a year, so these are divided by 8760, the number of hours in a year, to calculate an average hourly consumption rate. Histograms are created from same-sized households, with bins for every 0.1 kW increment of demand, as shown in Figure 4.2. These histograms show relative compositions for household demand values.

These histogram bins represent composition, so they were further divided by the total number of same-sized household to give percentage of occurrence at each kW value for a given number of household members. Figure 4.3 shows each different sized household composition normalized to produce a total combined percentage of 100 for each bin value. Relating the number of household members to number of vehicles is done using a U.S Census data table, shown in Table 4.1. These relations between electrical demand and the number

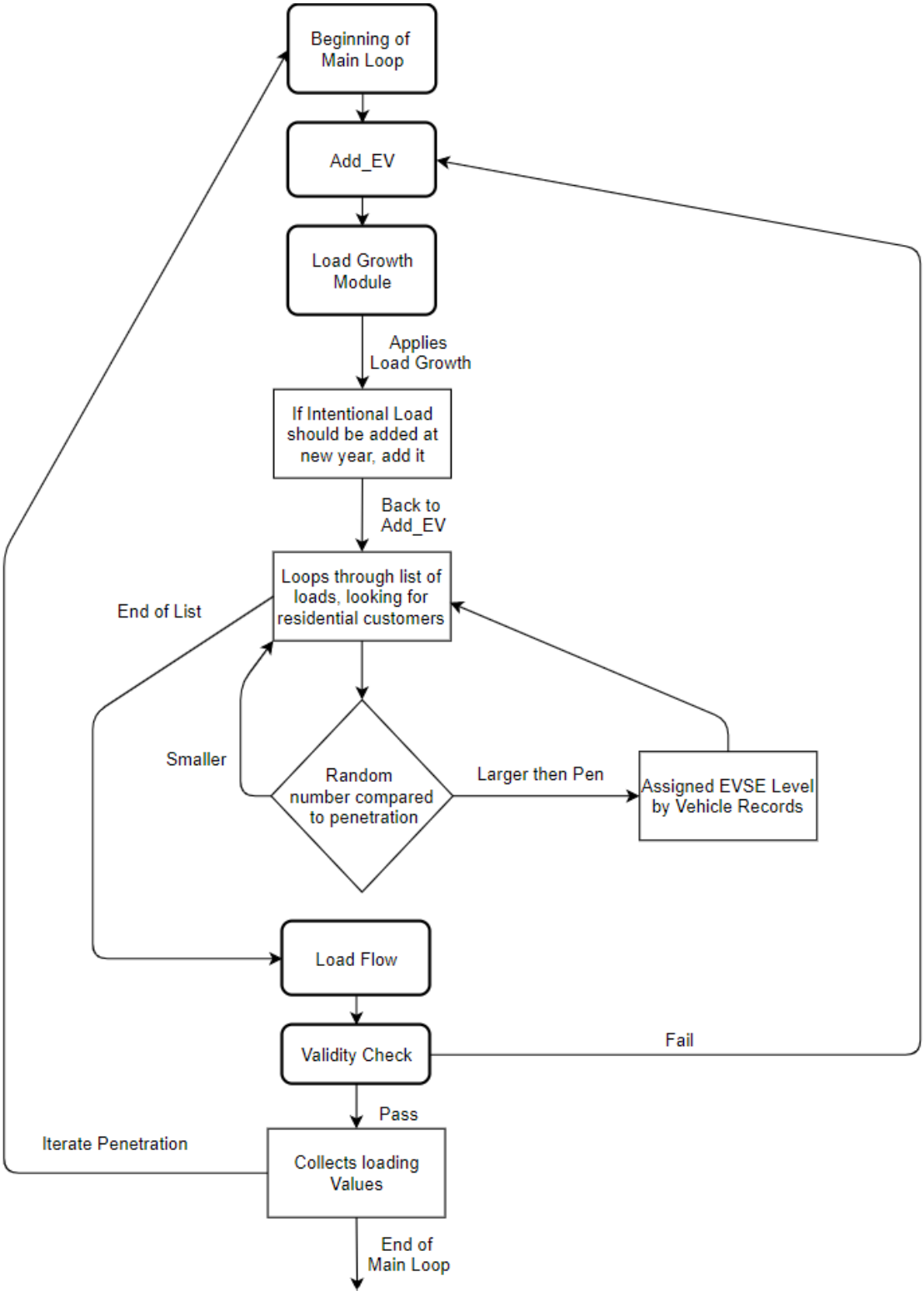

Figure 4.1: Flowchart of EV application loop.

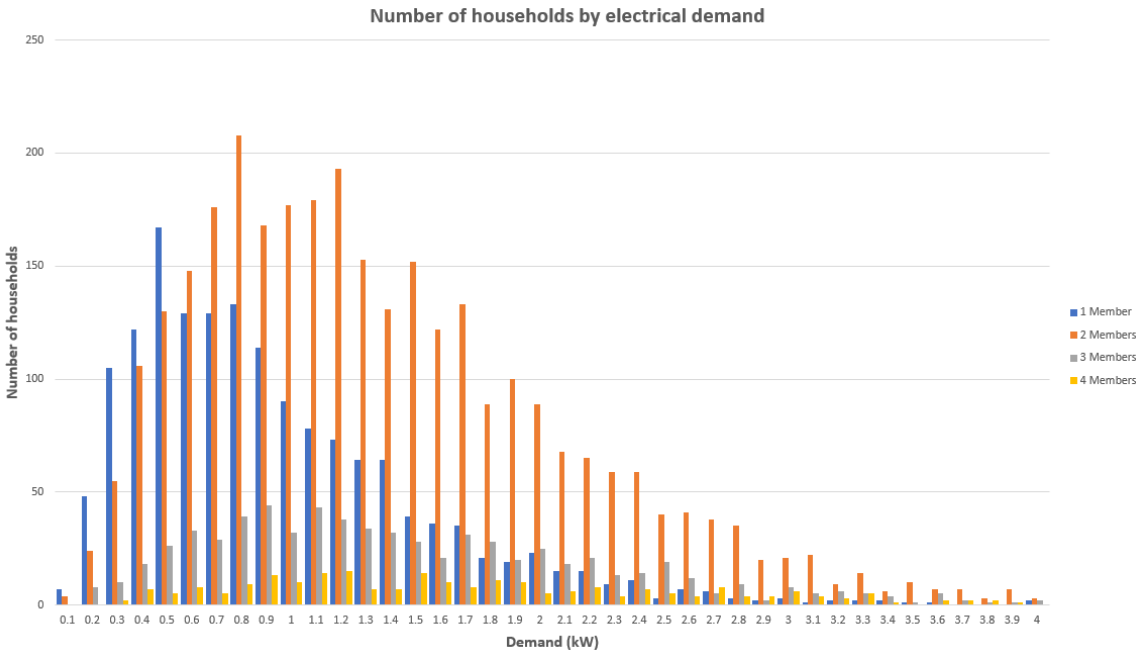

Figure 4.2: Composition of each household member size by demand.

of vehicles per household are provided along with the tool in a CSV format. Operators can supply different compositions. In order to realize this, the function *HouseholdVehicles* is used, which pulls each of the columns representing composition from a CSV file, and uses random number generators from 0 to 100. These random numbers determine the number of household members, then the number of vehicles available to the household from provided compositions.

The demand is truncated to a single decimal point for compatibility with the EIA compositions derived from histograms with bins of 0.1 kW. The number of ICE vehicles for each household is decremented for each EV added.

Label	United States	
	Estimate	Margin of Error (+/-)
Total:	120,062,818	161,148
No vehicle available	10,295,601	55,102
1 vehicle available	39,206,708	82,974
2 vehicles available	44,754,457	121,977
3 vehicles available	17,518,873	70,375
4 or more vehicle available	8,287,179	43,057
1-person household:	33,512,155	97,864
No vehicle available	6,172,106	43,200
1 vehicle available	21,976,907	84,710
2 vehicles available	4,262,642	37,136
3 vehicles available	798,440	14,332
4 or more vehicle available	303,060	9,821
2-person household:	40,993,693	118,919
No vehicle available	2,182,328	24,964
1 vehicle available	9,673,937	49,732
2 vehicles available	21,850,797	87,356
3 vehicles available	5,541,766	40,321
4 or more vehicle available	1,744,865	20,580
3-person household:	18,559,969	74,170
No vehicle available	906,637	15,831
1 vehicle available	3,748,951	31,957
2 vehicles available	7,329,742	46,886
3 vehicles available	4,946,000	39,038
4 or more vehicle available	1,628,639	17,585
4-or-more-person household:	26,996,001	75,633
No vehicle available	1,034,530	17,067
1 vehicle available	3,806,913	34,791
2 vehicles available	11,311,276	50,687
3 vehicles available	6,232,667	33,005
4 or more vehicle available	4,610,615	32,576

Table 4.1: Composition of Vehicles by Household Members [37].

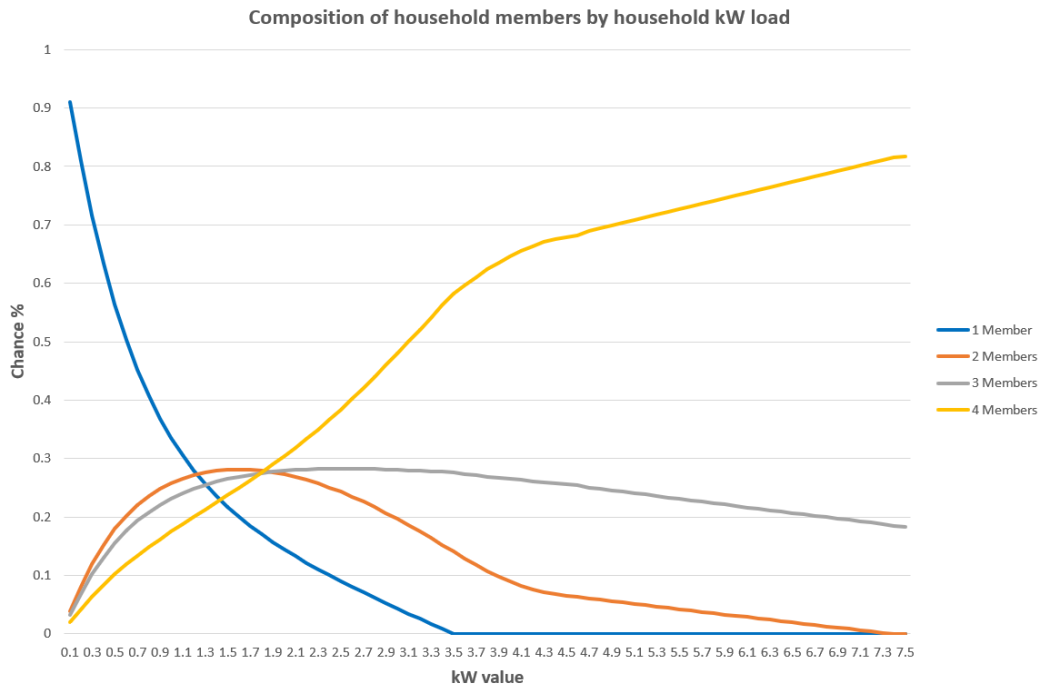

Figure 4.3: Composition of household members by demand.

4.2 Intentional EVSE Tool

Hi-P EVSE are applied to the distribution system through the *Intentional EVSE Tool*. This tool allows application of specific EVSE loads while considering the diversity factor for a site-specific EVSE transformer. It also provides the ability to delay the application of a Hi-P EVSE load to the distribution system. Overall, this tool allows Hi-P EVSE installations at any year, while creating a branch connecting the EVSE to the distribution system.

4.2.1 Hi-P EVSE Loads

Hi-P EVSE installations are handled in a separate function before stochastic EVSE application. Installation decisions are handled by user prompts, placing large EV loads at a spotload specified by the operator. As Hi-P EVSE projects have high power demand,

these loads are usually connected through a site-specific transformer. The *Intentional EVSE Tool* creates a similar asset branch to the specified spotload appropriate to the Hi-P EVSE demand. This new branch recreates the assets and nodes connecting the specified spotload to the distribution system with modified names. Adjusting existing asset names prevents creating an asset with a pre-existing name, which can lead to errors. The diversity factor is applied to the transformer rating before placement depending on the number of similar EVSE. This factor is applied to the demand of each class of EVSE individually in order to create an adequately-sized transformer. Figure 4.4 contains a flowchart view of the decisions for adding intentional loads.

4.2.2 Planned Hi-P Loads

In order to allow for Hi-P load growth, the CYME load growth module provides an asset year, which is compared to the list of planned Hi-P loads. When the year from load growth matches or exceeds the Hi-P EVSE commission year, the EVSE is applied to the distribution network. Planned Hi-P loads may be applied to any year between the first and last projected year of the EV penetration forecast.

4.3 System Growth Tool

The *System Growth Tool* ensures that the distribution system includes EV load growth as well as non-EV load growth appropriate to the system year. EV load growth is controlled through the percentage of remaining vehicles that need to be applied to represent an EV penetration change. Non-EV load growth is handled through the CYME load growth module,

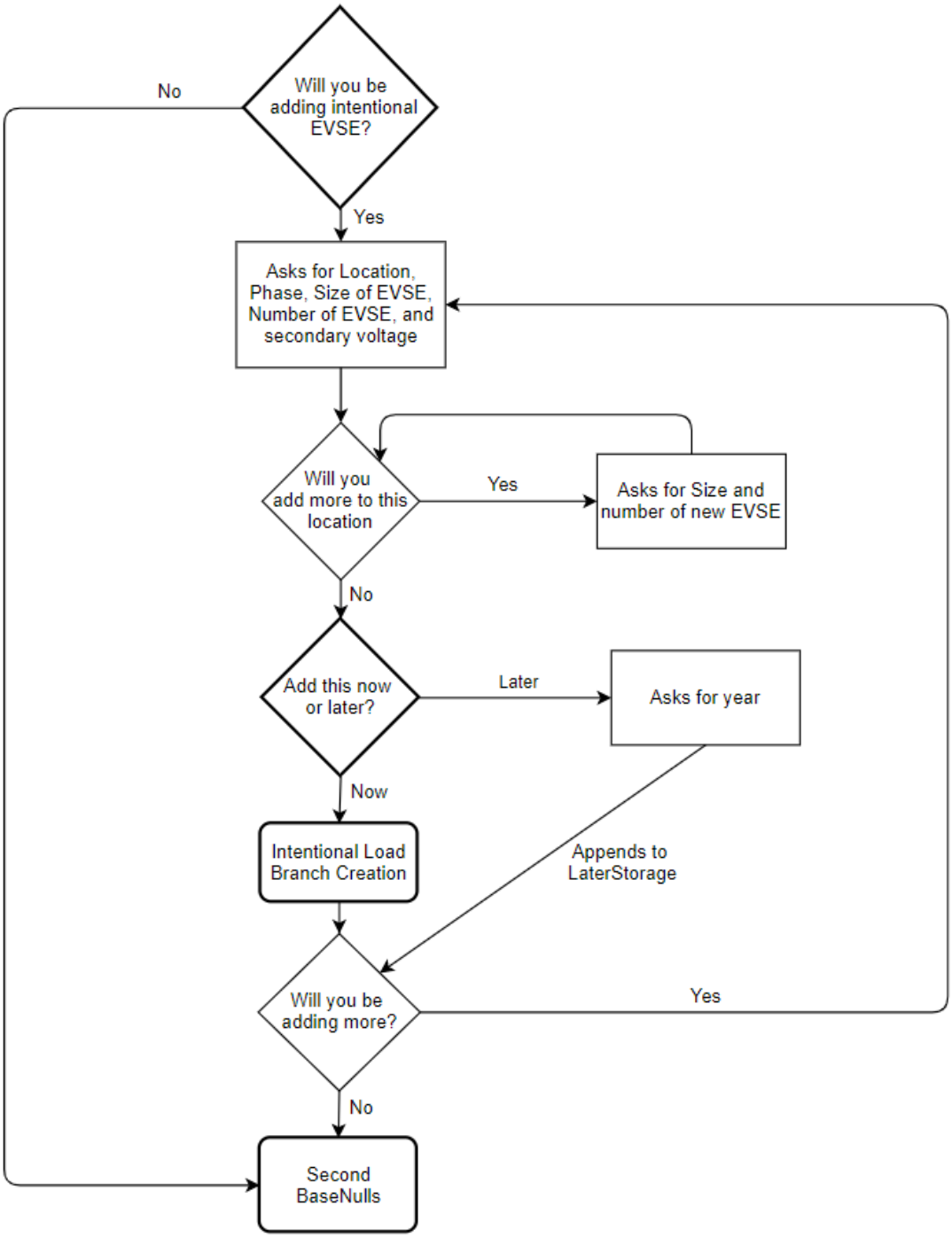

Figure 4.4: Flowchart of the User Decision Tree for Adding Intentional Loads.

which allows assets to have a current year, and increment the non-EV demand to reflect electrification. This tool allows non-EVSE load growth to be related to EVSE penetration forecasts.

4.3.1 Residential EV Penetration Increases

As EV penetration increases, the number of available non-EV vehicles decreases. In order to accurately represent EV penetration, the percentage of vehicles added is the step size of EV penetration divided by 100% minus the previously reached EV penetration. The first method, using Equation 1.1, is for increasing the penetration on a study that begins with no knowledge of current EV penetration. In the case of a study that understands the EV penetration, but has no way of separating the EV loads from the household loads, Equation 1.2 is used. This method allows the user to add EVs at a reduced rate to allow 100% penetration by only adding the available EV loads, removing the initial EV penetration from the denominator.

$$VehiclePercentage = \frac{(PenetrationStep)}{(100 - PreviousPenetration)} \tag{1.1}$$

$$VehiclePercentage = \frac{(PenetrationStep)}{(100 - PreviousPenetration - StartingPenetration)} \tag{1.2}$$

4.3.2 Non-EV Load Growth

Distribution system non-EV load growth is controlled via the current EV penetration. To allow this, the operator is required to provide a CSV with the expected EV forecast at

specific years. Each EV penetration is associated with a single year. In the case of EV penetration being between two values supplied, linear interpolation is used to calculate the estimated year. This year is then supplied to the CYME *Load Growth* module, which increments non-EVSE loads by a specified percentage to represent a new year on the system. If fewer values are provided, accuracy decreases due to the linear nature of the interpolation, so it is recommended to use a differential of less then four percent between penetration estimations.

4.3.2.1 EVSE

The CYME *Load Growth* module may exclude a customer type while applying load growth. EVSE are stored as an independent customer at each spotload created from this tool. The customer type, which may be specified by the operator, is used to denote each EVSE customer. This ensures that all EVSE growth occurs from new EVSE being installed, as opposed to non-EV load growth applied to EVSE.

4.4 Time Series Tool

Having time series functionality, the ability to examine assets at certain time steps, is pertinent to identifying overloading events. The *Time Series Tool* allows each vehicle applied to the study to use its own demand profile. Records were created to maintain consistency of EV application for each household. This tool allows time-series demand profiles to be applied and their impacts observed.

4.4.1 EVSE Charge Profiles

EVSE demand profiles are stored in a way that makes indexing straightforward. For residential EVSE indexes are attached to the records for each vehicle. One CSV for each Level 1 and Level 2 EVSE demand profiles are supplied, which are accessed depending on the vehicle EVSE level. Intentional EVSE demand profiles are designed for each planned EVSE. These demand profiles are assigned to the EVSE in the order they are input during *IntentionalLoad*. For each time step, the demand for each EV are accessed by profile index, current time step, and designated EVSE level.

4.4.2 Time Series Changes

To find the loading and voltage impacts during a certain time period, two functions were created. The first function, *Reapply*, applies the new demand profile time step for each EVSE. This function allows this suite of tools to simulate the distribution study for each EVSE demand profile time step. The second function, *TimeFlow*, collects data from each asset related to loading and voltage level. Each loop appends the recorded quantities to a list containing recorded data at each time step. These functions work together to simulate each time step and record the EVSE impacts on assets.

4.4.3 Information Storage

After the number of vehicles available to a household is determined, records are created for each vehicle. These records includes the index of EV demand profile, if the vehicle has been applied, and what level EVSE to apply. Random number generation is used to find the

demand profile for each vehicle. The level of EVSE for each vehicle is determined from the composition of Level 1 and Level 2 EVSE. When a vehicle is linked to its household, the number of available vehicles decrements by one, and the application records are changed depending on which vehicle was added. These records ensure each vehicle cannot be added twice, and their demand profiles are consistent between time steps.

4.5 Data Collection Tool

The *Data Collection Tool* is used for identifying loading and voltage data, as well as ensuring the data are accessible. This tool identifies the different phase loading and voltage levels for each asset. The EVSE impact data are stored using a method that allows simple loops through each EV penetration and time step. These qualities ensure validity of phase loading and voltage as well as ease of accessing the different time steps.

4.5.1 Asset Loading and Voltage Collection

Assets being examined by this tool include transformers, transformers-by-phase, and distribution lines. Transformers and distribution lines can have loading and voltage values directly accessed from CYMDist. Individual phase information is gathered for a more comprehensive representation of asset loading. Transformers-by-phase were disconnected within the Ceder Hills CYME study used to test this suite of tool. Transformer-by-phase voltage levels can be accessed directly, but loading information must be derived from the power flowing through the asset and nominal power for each phase loading. These derived loading values are within 0.5% of actual loading values, due to not considering losses inside

the transformer. This allows information from the different phases of each asset to be recorded even if the asset itself is not in a connected status in CYME. Loading and voltage information are stored in two formats, every asset's value, a certain number of the worst loaded assets decided by the use.

4.5.2 Stored Loading and Voltage Format

Loading and voltage records are stored in tuples containing the asset name as well as the three phase values. These records are maintained in separate variables. Each quality is stored into lists covering entire time periods, which is further appended into a list of each EV penetration. These variables can be looped through to access each penetration, and access each time step in order of occurrence.

5 Validation

The tools in this thesis invoke multiple functions that work together to provide tool functionality. This section presents the validation of each of these tools. Each case presents an input-output diagram for the function. Validation is important for demonstrating that the tools perform as designed. These validation cases are presented per their related tool.

5.1 Stochastic Residential EV Tool

The *Stochastic Residential EV Tool* is used to find the number of vehicles at each household and apply EVSE load demand at the household when EVs increase. The functions *HouseholdVehicles* and *Add_EV* are tested to ensure they produce the expected results. *HouseholdVehicles* shows how which steps are taken for each decision of households members and number of vehicles. The *Add_EV* function is tested by comparing CYME studies from the first EVSE addition. These functions allow EV penetration to be represented on the distribution system.

5.1.1 Household Vehicles

The *HouseholdVehicles* function uses data compositions from the EIA and US Census. These data reflects the relationship between household demand to number of household vehicles. The inputs and output of this function are shown in Figure 5.1. Validation of this

tool is done by analyzing each step converting household demand to vehicles. Figure 5.2 shows this information, for five households. The red underlined value for each line represent the household name and its demand binned to 0.1 kW. The blue underlined values include three values. These include the random number used to convert between demand and household member, the composition relating the two, and the result. This composition represents the chance of have one, two, three, or four household members. For example the first household's composition is [33.5,59.3,82.5], there is a 33.5% chance of a one household member, a 25.8% chance of two members, 23.3% chance of three members, and 17.5% chance of four members. The composition used is related to the household demand, identical blue underlined compositions can be observed between the first and second household and third and fourth household, as they have identical household demand. The green underlined values contain similar information to the blue values, but for converting from household members to number of vehicles. This second sets composition includes an additional value to the blue composition, for no vehicles. These green compositions are determined from the number of household members, as you can notice for the first and third household. Each composition and the values used to access have been determined as accurate, validating this function.

The compositions of household demand to members are accessed depending on the level of demand. Each household receives different member compositions depending on electrical demand. Three instances of the two households with matching demand are included in Figure 5.2. These show that compositions are not decided at random, as household with

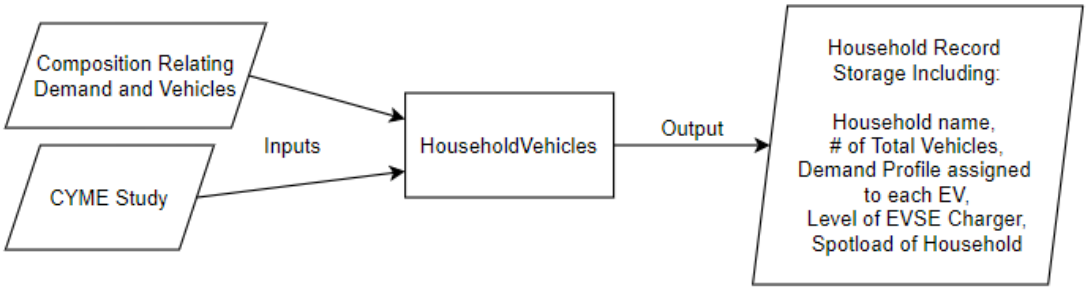

Figure 5.1: Inputs and outputs for the HouseholdVehicles function.

Household Number	*Demand*	*Random Number 1*	*Composition Household demand to members*	*Household Members*	*Random Number 2*	*Composition Household members to vehicles*	*Household Vehicles*
[['8860466753'],	9,	46.7,	[33.5, 59.3, 82.5],	2,	33.4,	[5.3, 28.9, 82.2, 95.7],	2]
[['8550901459'],	9,	2.8,	[33.5, 59.3, 82.5],	1,	51.5,	[18.4, 84.0, 96.7, 99.1],	1]
[['9250844428'],	14,	42.0,	[21.7, 49.8, 76.3],	2,	86.0,	[5.3, 28.9, 82.2, 95.7],	3]
[['1960466756'],	14,	18.3,	[21.7, 49.8, 76.3],	1,	71.7,	[18.4, 84.0, 96.7, 99.1],	1]
[['9860466754'],	3,	9.2,	[63.4, 78.6, 91.6],	1,	74.1,	[18.4, 84.0, 96.7, 99.1],	1]

Figure 5.2: HouseholdVehicles use case showing household member and vehicle determinations.

the same demand receive the same compositions. As mentioned in the previous paragraph household members compositions only have four possible compositions, one, two, three, or four members. Compositions for vehicles include five possibilities, zero through four.

5.1.2 Add_EV

Add_EV is the function used for adding EVSE stochastically into the distribution study. Inputs and outputs for this function are shown in Figure 5.3. This function is tested by using it to place an EVSE then showing the results. A study is saved directly after the first EVSE application. The spotload changes is used to locate the modified household along with the EVSE demand.

Table 5.1 shows the base spotload and the spotload after *Add_EV*. A demand of 6.6

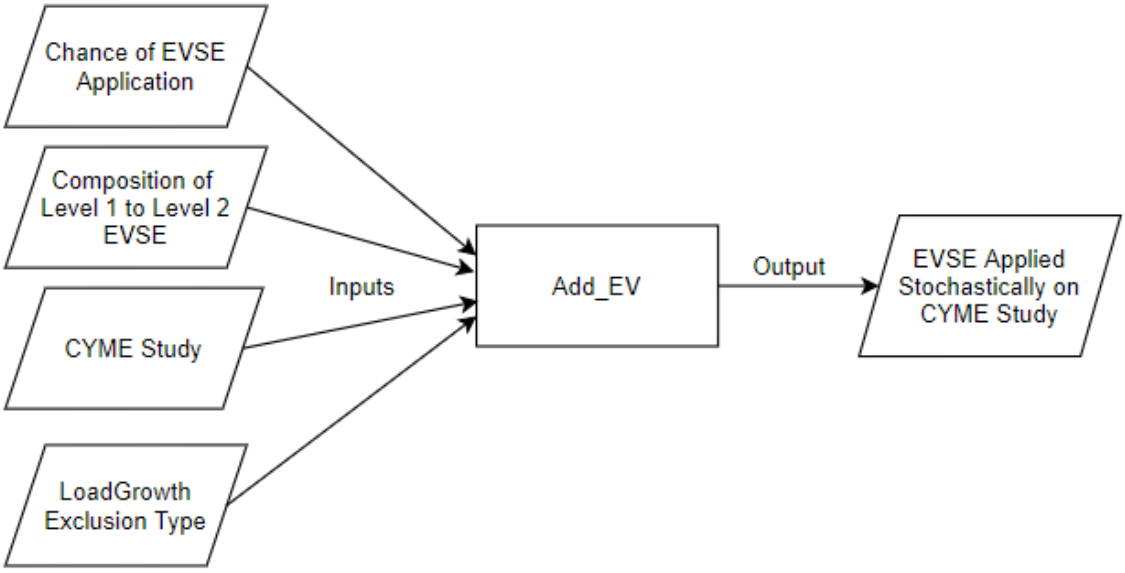

Figure 5.3: Inputs and outputs for the Add_EV function.

Pre-Tool Spotload			
Customer Number	**Customer Type**	**Year**	**Actual kW**
7100619508	Residential	2020	1.93
Post-Tool Spotload - One EVSE Applied			
Customer Number	**Customer Type**	**Year**	**Actual kW**
7100619508	Residential	2026	2.11
OID_1147000	Fixed	2026	6.6

Table 5.1: Spotload Customer information before and after Add_EV.

kW was assigned to the EVSE for ease of identification. The additional 'customer', named *OID_1147000* is the customer for EVSE storage which is created in every spotload with an initial demand of 0 kW. As the demand of *OID_1147000* is equal to the assigned demand of 6.6 kW, this EVSE was added correctly to the distribution study.

5.2 Intentional EVSE Tool

The *Intentional EVSE Tool* is used to place non-stochastic EVSE onto the distribution system. Specifically the *IntLoadBranchCreation* function is used when adding these EVSE loads to the system. This function gathers the assets connecting the reference spotload to the distribution network and creates appropriate assets mirroring the branch. This function enables the *Intentional EVSE Tool* to place EVSE independent of the current distribution study assets.

5.2.1 Intentional Branch Creation

This *IntLoadBranchCreation* function takes user inputs and produces changes in the study model. Inputs for this function include the reference location and other user inputs used to output a distribution study with intentional EVSE. These inputs and outputs are shown in Figure 5.4. In order to demonstrate the functionality of this tool, schematics were copied of the system before and after adding an EVSE. Figure 5.5 show an additional transformer and spotload from a CYME study. Additionally, the EVSE spotload transformer rated phases are shown in Figure 5.7, with the spotload demand itself shown in Figure 5.6.

5.3 System Growth Tool

Distribution system growth is handled through the *System Growth Tool*. This tool ensures that demand increases due to EV penetration growth are matched with distribution system

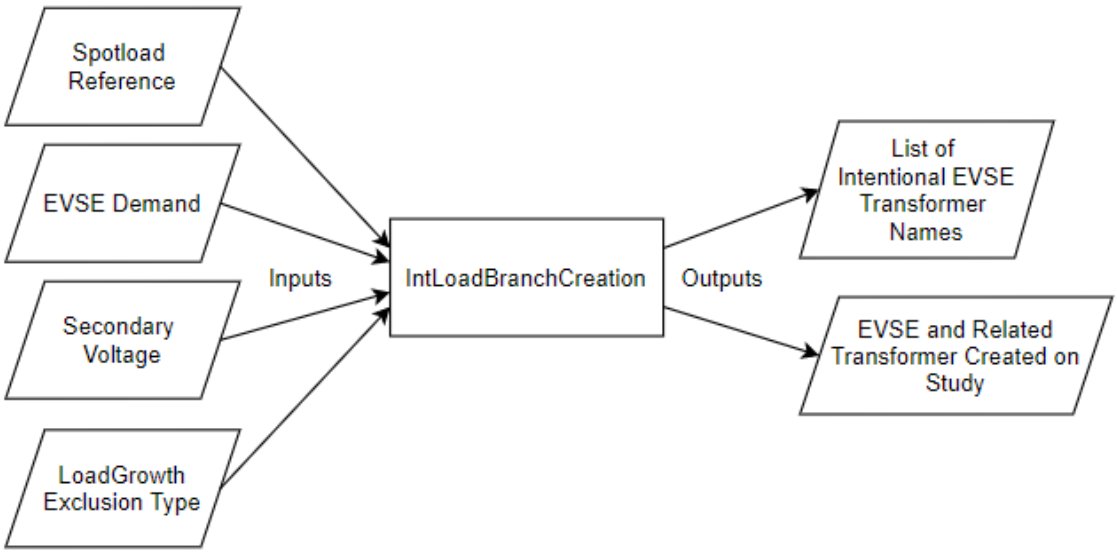

Figure 5.4: Inputs and outputs for the IntLoadBranchCreation function.

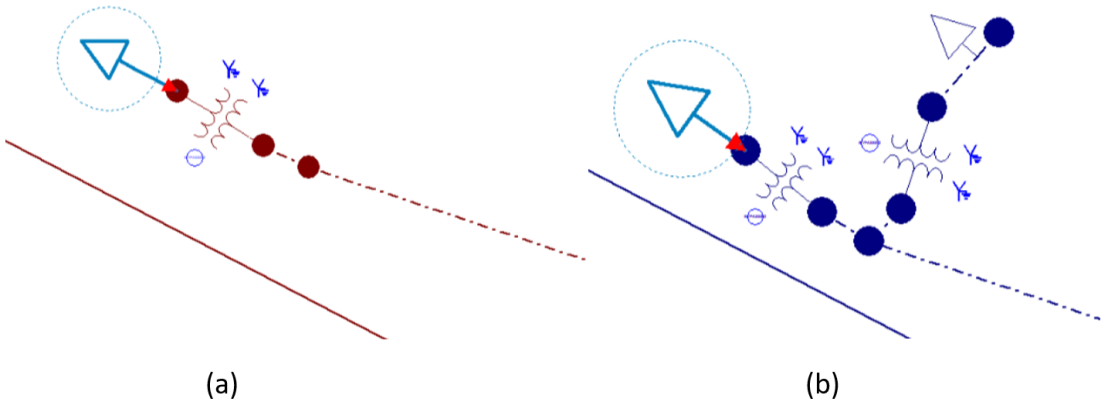

Figure 5.5: CYME distribution system before (a) and after (b) an IntLoadBranchCreation application.

electrification. In order to enable this, EV penetrations must be connected to a given year. The function *PenetrationVsYears* is used to gather the year EV penetration occurred.

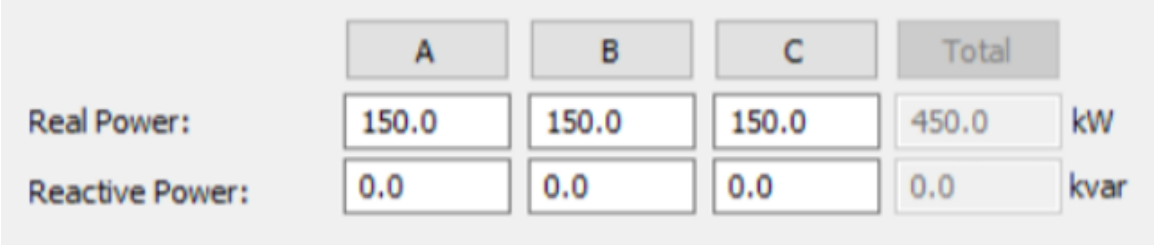

Figure 5.6: Demand properties from spotload created using the IntLoadBranchCreation function.

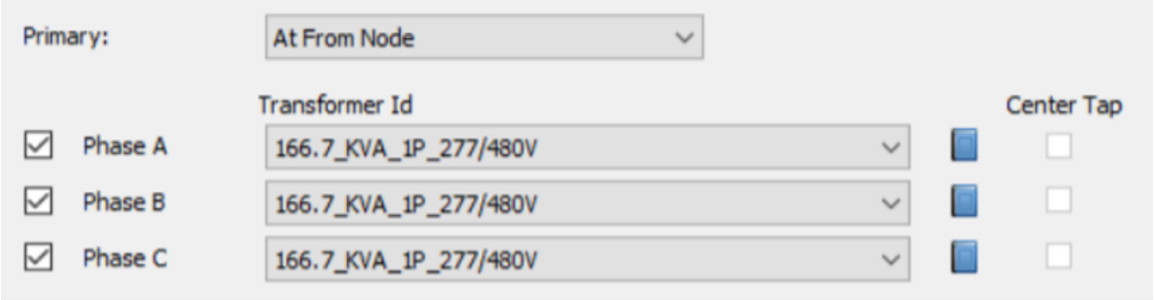

Figure 5.7: Phase rating properties from transformer created using the IntLoadBranchCreation function.

5.3.1 Penetration Vs Years

This function *PenetrationVsYears* is used to make sure the current distribution system matches the simulated year. Inputs and outputs for this function are shown in Figure 5.8. The years gathered by the function are compared to the original EV forcast. Figure 5.9 showcases the difference between the forecasted year, and the year this tool applies. The difference between forecasted year and applied year is reduced for a forecast with a larger dataset.

5.4 Time Series Tool

The *Time Series Tool* was created to apply demand profiles and allow time series analysis. This requires a function that can apply new profile demands without adding additional EVSE. The function *Reapply* was created to meet this requirement. This function allows changes

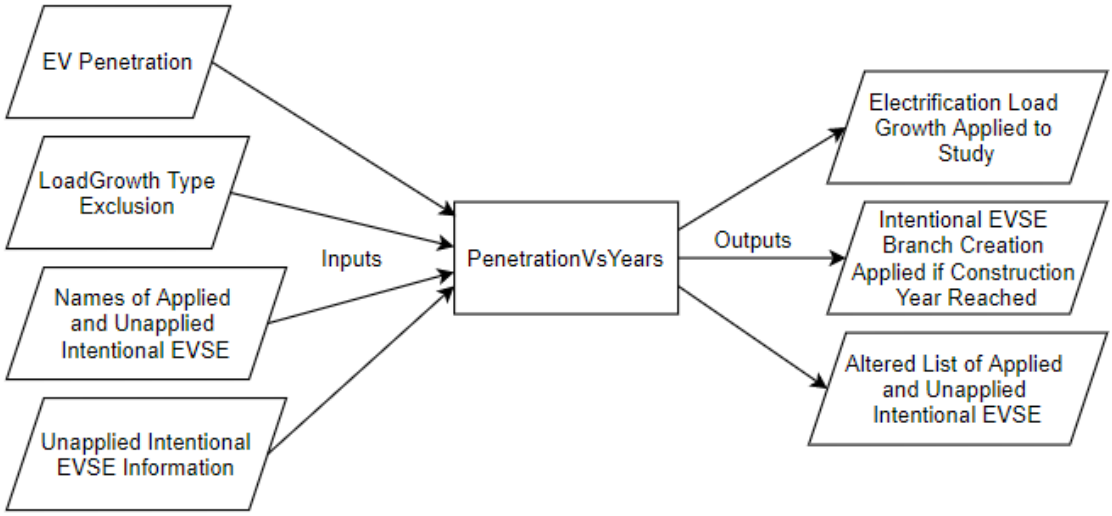

Figure 5.8: Inputs and outputs for the PenetrationVsYears function.

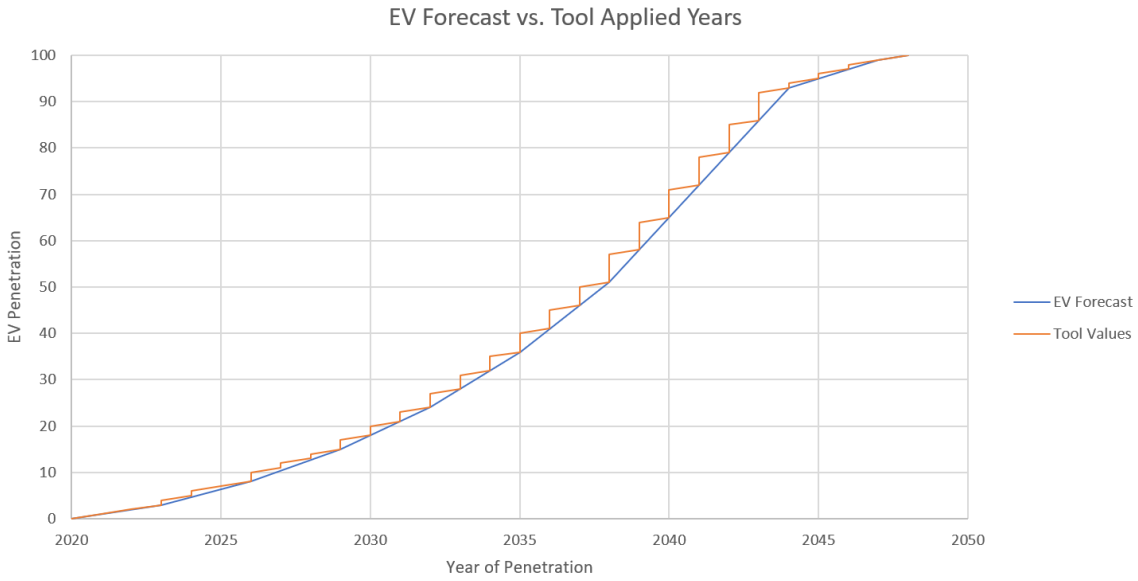

Figure 5.9: Equipment year, forecast EV years vs. applied load growth years.

to the distribution study through household records, only modifying study demand when EVSE demand changes.

5.4.1 Reapply

Reapply is the function that changes EVSE demand based on the current time step of a demand profile. The inputs and output of this function are shown in Figure 5.10. This function deals with changing EVSE that have already been applied to the study. This function is tested by comparing a list of EVSE customers applied in *Add_EV* to a list of EVSE customers changed during *Reapply*. It is found these lists are identical. Because of the nature of the test there is no graphical representation.

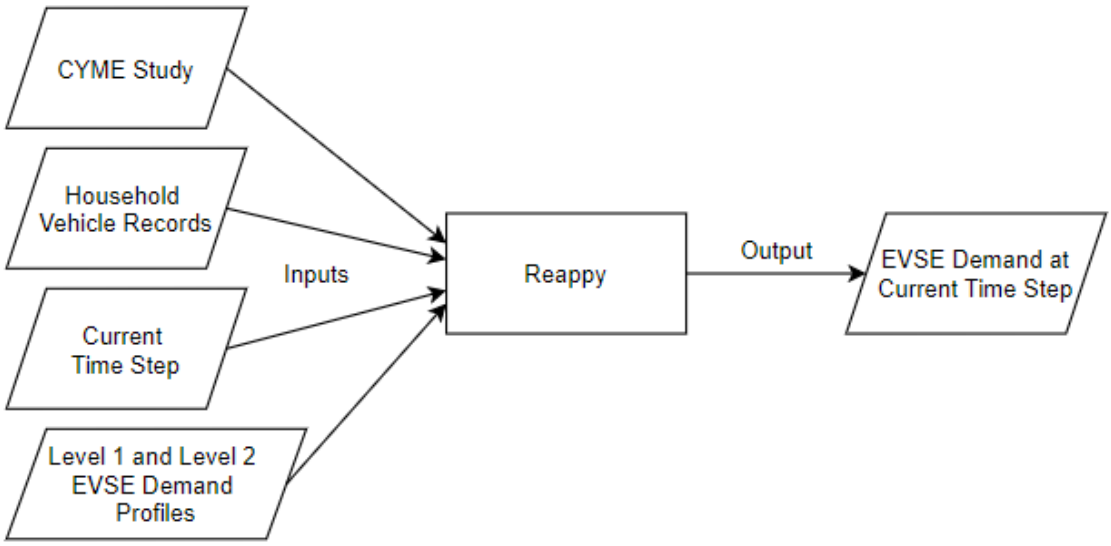

Figure 5.10: Inputs and outputs for the Reapply function.

5.5 Data Collection Tool

This suite of tools collects a large amount of information. In order to process these data, the *Data Collection Tool* was created. The most important functions for this tool are *Processing* and *CSVOutputs*. *Processing* is used to created formatted storage for *CSVOutputs*. The

function *CSVOutputs* outputs this formatted data into CSVs and generated plots. These two functions work together to output the resulting documents.

5.5.1 Processing

The function *Processing* is used to store desired information in a format used by *CSVOutputs*. This involves ordering by loading and voltage level data individually. Length of overloaded assets are used to create binned histograms by half hour using the length of events. The consistency of these are checked through comparing the order of the values to the list of worst asset over loading or under voltage. In addition, the values of over loaded assets used to create histograms are compared to the formatted histogram output to verify accurate binning. Inputs and outputs for function *Processing* are shown in Figure 5.11.

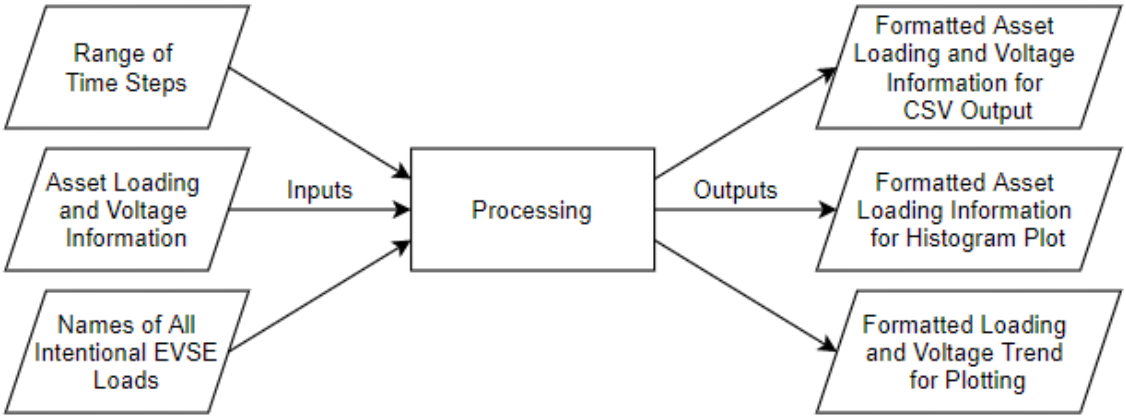

Figure 5.11: Inputs and outputs for the Processing function.

5.5.2 CSVOutputs

CSVOutputs creates the output documents for this suite of tools. The inputs and outputs for this function are shown in Figure 5.12. The function uses the formatted information from *Processing* to output asset data into CSVs, histograms, and trend graphs. This function is tested by supplying the loading and voltage information and making sure the output files are correct. Each asset is given the same set of loading and voltage information, which creates the same output file, with different asset names as expected.

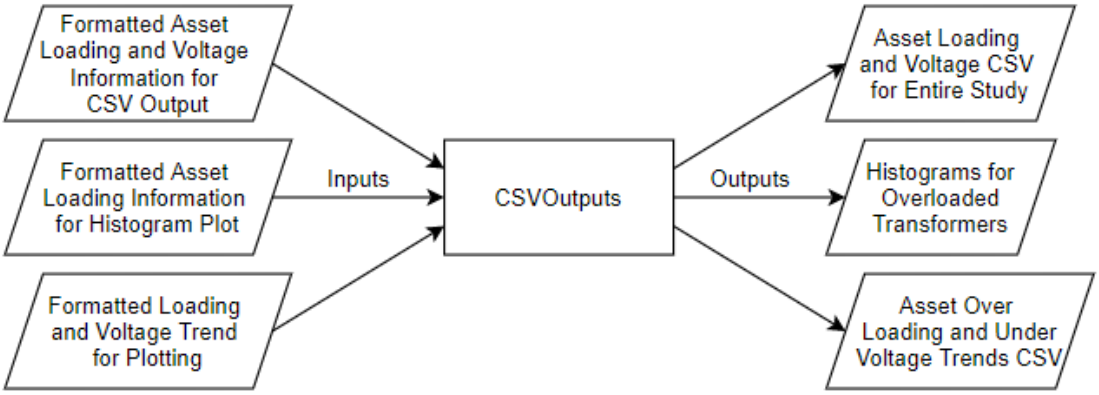

Figure 5.12: Inputs and outputs for the CSVOutputs function.

6 Discussion

The following sections present an analysis of tool efficacy within the context of a case study. Voltage and loading are examined. Analysis is performed to understand impacts and trends of loading and voltage values due to increased EVSE penetration. After the test case analysis, the capabilities of this suite of tools are evaluated. At the end a description of this tools required files is provided along with where they can be found.

6.1 Suite of Tools Test Case

The suite of tools created for this thesis work together to simulate distribution systems affected by EVs. A test case study of a distribution feeder was conducted to demonstrate tool capabilities when working in tandem to provide data for analysis. The conditions include an EV penetration ranging from 10% to 90% with a step size of 20%, and a time range of 3 AM to 9 PM for the demand profiles. Additionally, the study includes four intentional high power EVSE, one added right away, the remaining three EVSE set to be install during the years 2030, 2036, and 2042. This range of intentional EVSE installation dates allows insight into when they are installed, and how uninstalled EVSE are represented during the study.

Case study results are analyzed for each of the different asset classes. CYME asset class *transformer* represents substation transformers. *Transformers-by-phase* are the asset class used for distribution transformers. These transformers-by-phase are examined for both

loading and voltage events per time step across the asset class. Both transformers-by-phase and distribution lines have a single asset examined for change in loading and voltage level across each EV penetration. This single asset information shows the impact of increased EVSE via loading changes, and shows the resulting affect on voltage level.

This test case was run an a power lab workstation (3.6 GHz i7-4790 CPU with 8 GB of RAM). With the workstation used each time step took roughly 12 seconds, with each EV penetration lasting up to 25 minutes. The total time this simulation including processing and outputting of information was 135 minutes. The simulations time step was limited to recording information every ten minutes. This is due to the EVSE demand profiles only offering charging demand in ten minute intervals.

6.1.1 Transformer-By-Phase General Loading and Voltage

Transformers-by-phase are the largest number of transformers within the study. These transformers include all but the distribution system substation transformers. Because of their large number and importance as the majority of transformers, their trends are used to make sure this suite of tools produces useful outputs. In Figure 6.1, the overloaded assets for each time step and each phase are shown. These overloaded time periods clearly show an increase in occurrence at higher EV penetrations. The EV penetration percentage is shown in the figure legend. One can notice across all phases there are no time steps without at least one overloaded asset. This is due to some assets being overloaded in the base study before EV penetrations are applied.

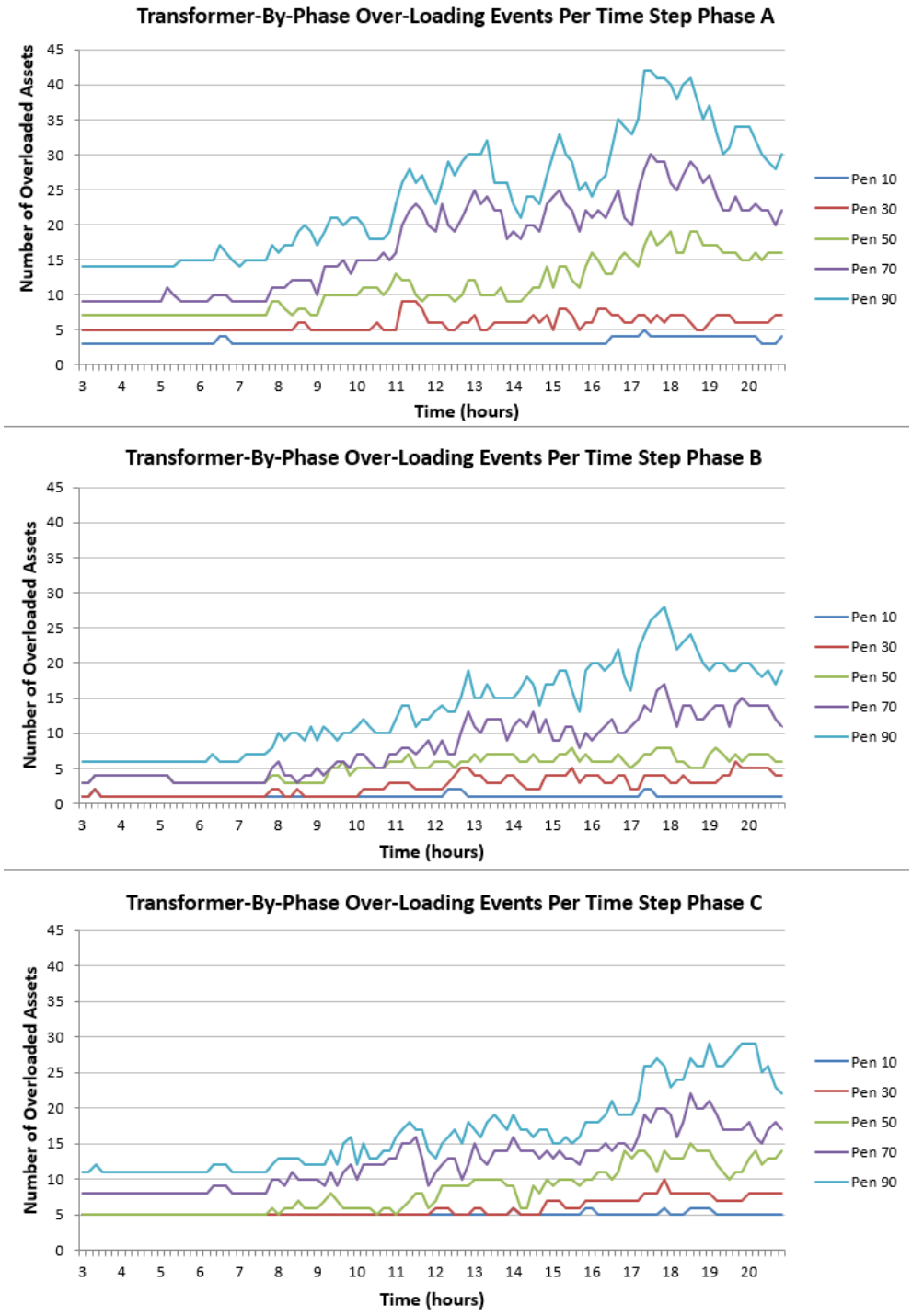

Figure 6.1: Transformer-by-phase overload occurrences as a function of time per EV penetration on phases A, B and C. The number of overloaded events increases with EV penetration.

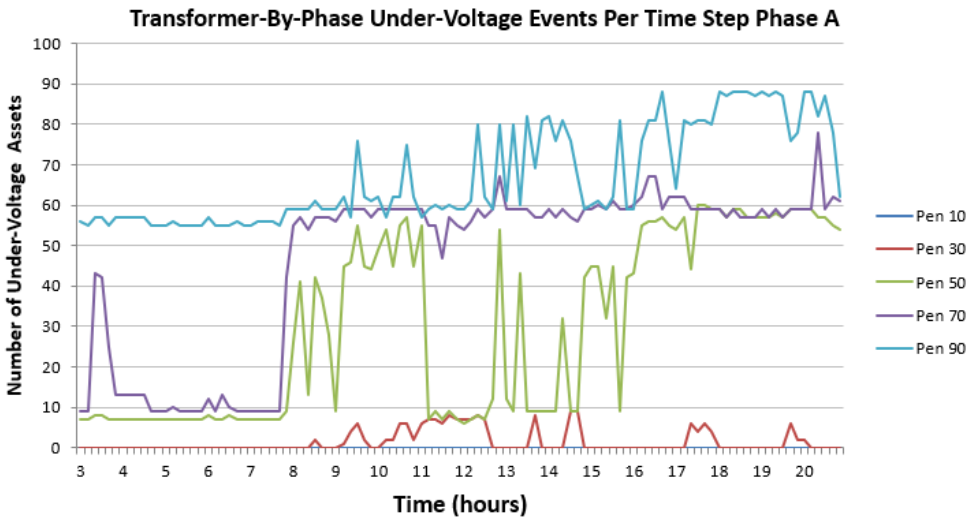

Figure 6.2: Transformer-by-phase under voltage time periods per EV penetration phase A. Under-Voltage events increase with EV penetration.

Under-voltage events are shown in Figure 6.2, which shows a single phase. This is the only phase that experiences under-voltage events. Other phases often have per unit voltages larger the 1.0. Voltage levels for this study of less then 0.95 per unit were rarely found. The graph in Figure 6.2 was produced using a threshold of 0.98 per unit voltage in order to validate the efficacy of the tool. Beyond 50% EV penetration, under-voltage events occur frequently due to transformers already under load being affected by increased EVSE demand.

6.1.1.1 Transformer-By-Phase Over-Loading Histogram

Transformer over-loading events vary in severity depending on the duration of the event. Long duration events may result in damage to the transformer and distribution lines if not cleared by relays. Histograms are presented in Figure 6.3, one plot for each electrical phase. These histograms have hour bins, starting at events lasting at least an hour and increasing

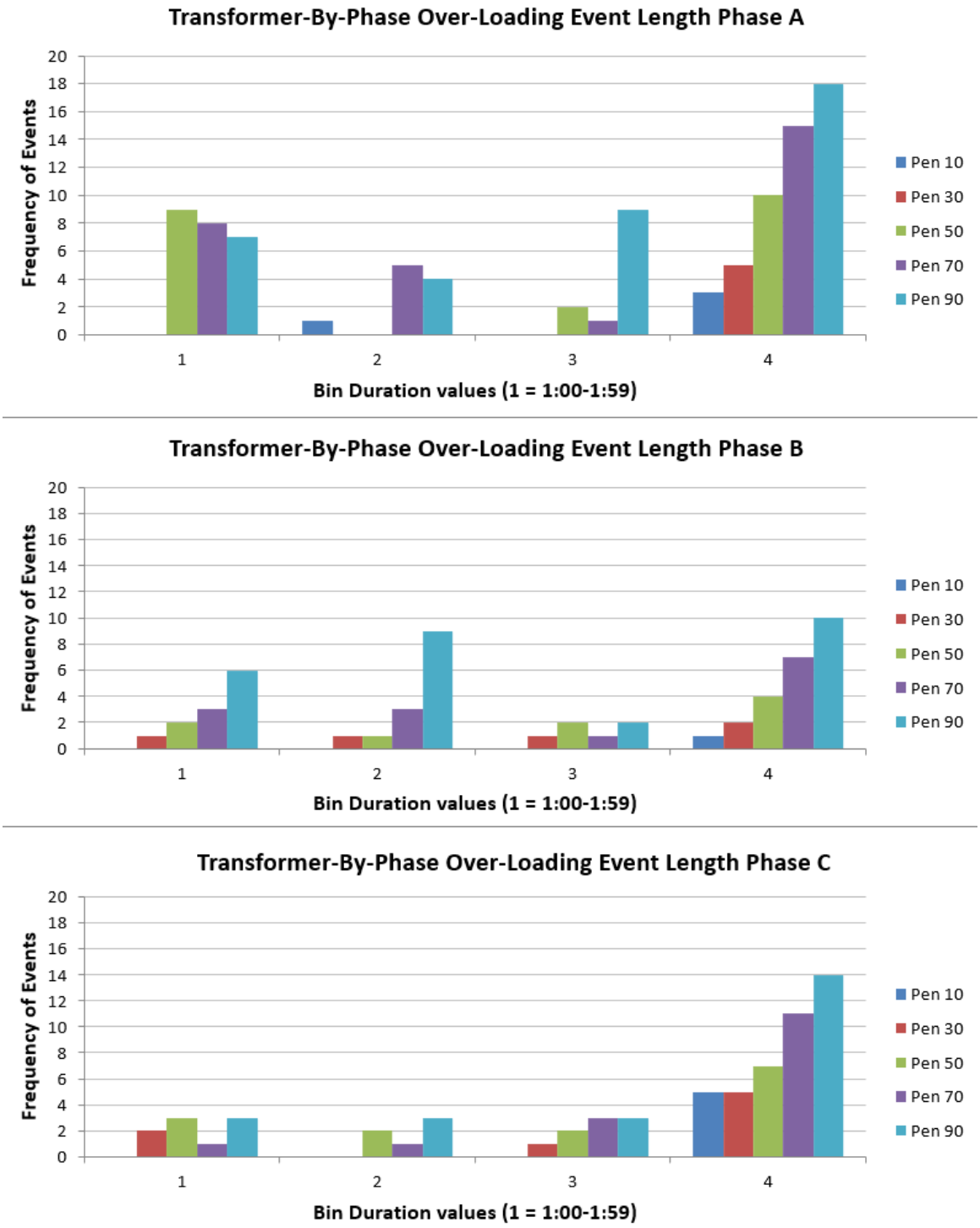

Figure 6.3: Transformer-by-phase loading duration histogram phase A. Over-Loading events increase with EV penetration.

by one hour for each bin. These histograms show that large EV penetrations such as 70% and 90% have more events in the 4th bin, representing at least 4 hour events. The smallest EV penetration, 10% has values in bin 4 due to a few constantly overloaded transformers from the base study. Overall there is an obvious trend on higher EV penetrations resulting in longer lasting over-loading events.

6.1.2 Location of Voltages Analyzed

For this test case analysis, two different voltages are considered. These two voltages are the voltage drop from source to asset, and the assets input voltage. The locations used to measure voltage are shown in Figure 6.4. Voltage levels used to calculate voltage drop were recorded directly after the regulating transformer, and directly before the transformer asset. The asset input voltage measurement is shown on the right of the figure. This shows the per unit phase voltage with respect to ground. Voltage start with a value of 115 kV at the substation transforming to 12.47 kV for the distribution network. This 12.47 kV rated voltage for the distribution network is then transformed to 0.42 kV for feeders.

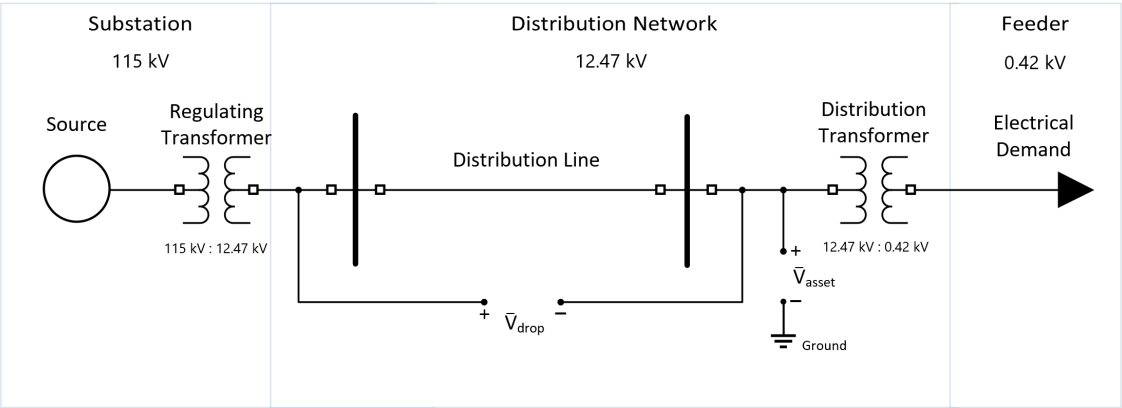

Figure 6.4: Distribution system voltage measuring Points for voltage drop and asset voltage.

6.1.3 Single Asset Voltage and Loading Information

The transformer asset used to showcase loading and voltage information is named A:25 36781. This asset is one of the most affected by overloaded events. It serves five households within the feeder. In Figure 6.5, one can notice changes in the profile of the transformer loading characteristics. The general loading increase across EV penetrations is due to electrification raising all non-EVSE demand for each incremental system year. Changes

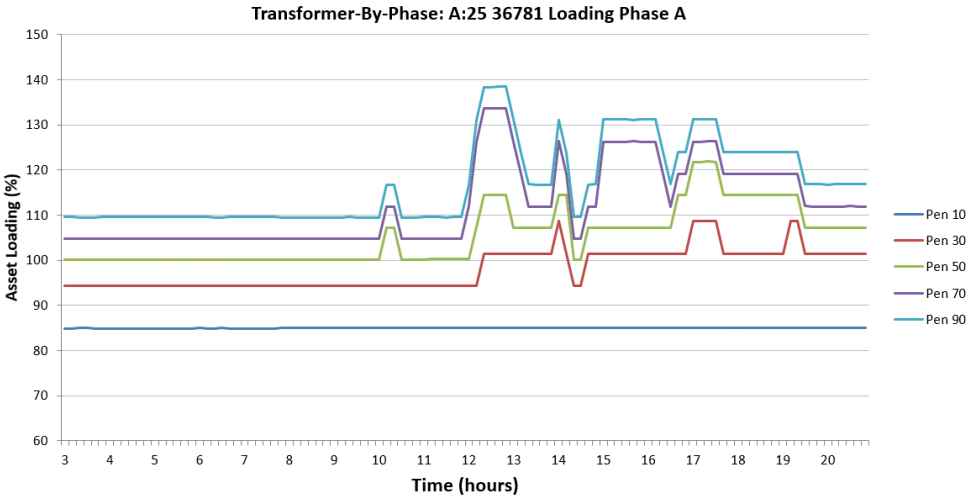

Figure 6.5: Transformer-by-phase loading values for A:25 36781. Loading increases with EV penetration increase in changes at high penetration due to multiple EVSE.

such as the spikes in loading percentage represent the EVSE demand increasing through addition of electrical demand. Some time periods saw as much as a 23% increase in asset loading from one EVSE penetration to the next, such as 10% to 30% EV penetration. This change in penetration reflects six years of change for the distribution system. This would suggest that these changes must be assessed with a smaller step between EV penetrations, but as once every six years is already unreasonable, this is understandable.

Changes in voltage levels, shown Figure 6.6, experience higher volatility with increases of EV penetration. These can be associated with the different EVSE demand profiles

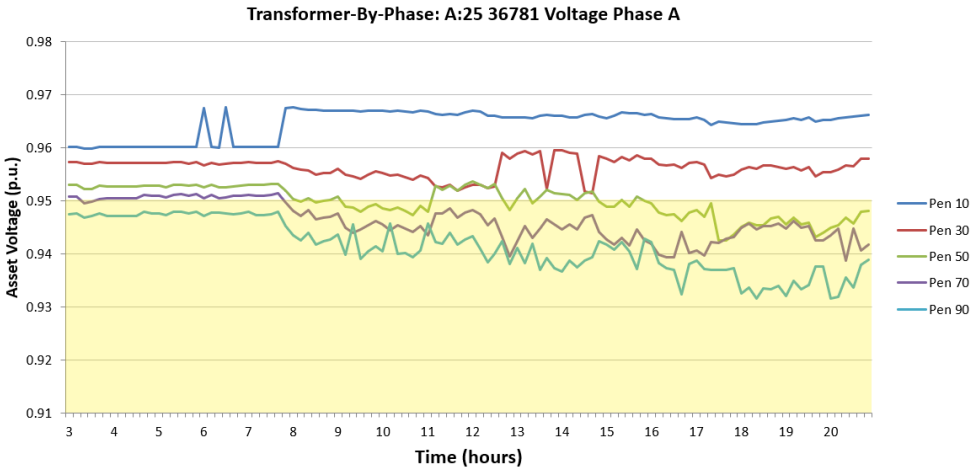

Figure 6.6: Transformer-by-phase voltage values for A:25 36781. Voltage at transformer decreases with higher EV penetrations. Voltage drops below 0.95 p.u. for high EV penetrations.

offering demand at different times. The voltage variation is small but not insignificant, with a respective maximum and minimum of 0.967 and 0.931 pu voltage. Voltage shows a consistent decrease with increased EV penetration. For EV penetrations of 50% and

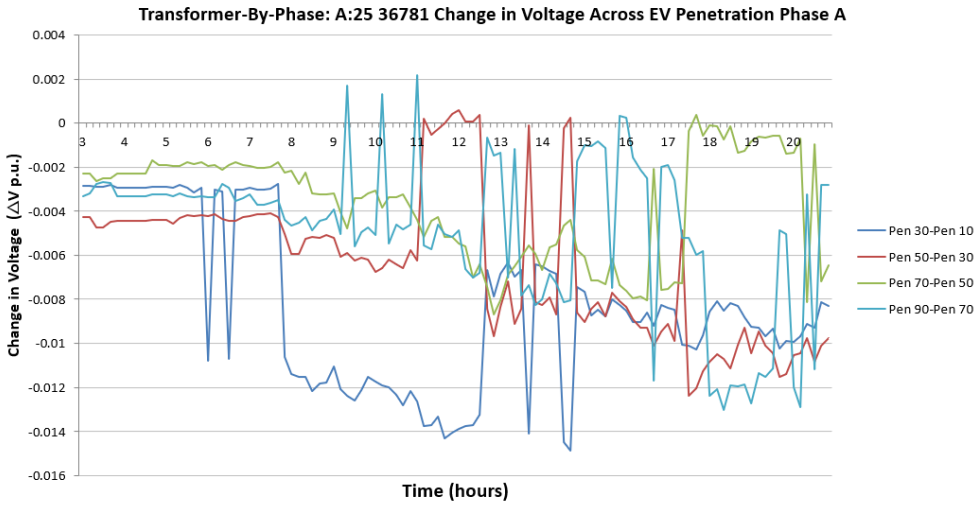

Figure 6.7: Transformer-by-phase voltage drop for A:25 36781 between EV penetration levels. Transformer voltages drop with increases in EV penetration.

above this asset experiences under voltage events with a per unit voltage less then 0.95. The differences in voltage between each penetration level are shown in Figure 6.7. As the values are mostly negative, this reflects the decreasing voltage in Figure 6.6.

6.1.3.1 Voltage Drop from Source to Asset

A distribution system experiencing demand and EV growth should show a decrease in transformers input voltage level. Despite this, many transformers show a voltage increase when looking solely at input voltage level. This happens due to regulating transformers at the substation; the distribution side source voltage levels increase with loading in order to keep voltages within ± 5% of nominal. This increase in voltage can make it difficult to observe voltage drops due to high demand. The voltage drop between the source and the asset A:25 36781 are shown in Figure 6.8. The voltage drop from source to asset increase with EV penetration. This figure show a clearer difference between each EV penetration

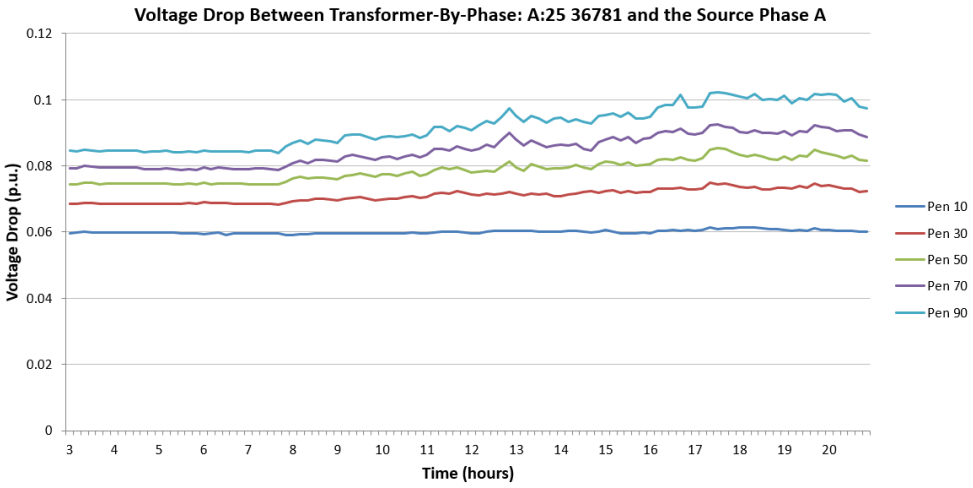

Figure 6.8: Voltage drop from source to transformer-by-phase asset A:25 36781 across EV Penetrations. Voltage drop increases as EV penetration increases.

when compared with Figure 6.6.

The transformer-by-phase A:25 36781 is one of the assets furthest from the substation. In contrast, transformer-by-phase asset B:25 27794 is close to the substation. Analysis of this asset shows less impact on voltage as expected because of regulating transformers raising the voltage level in the substation. This impact is shown in Figure 6.9. The input voltage to this asset raises with EV penetration due to a regulation transformer increasing the sources voltage to maintain stability. Input voltage levels seem to increase with EV penetration, when looking at the asset voltage plot. This asset experiences over-voltage events with 90% EV penetration. The voltage drop however shows an increase from source

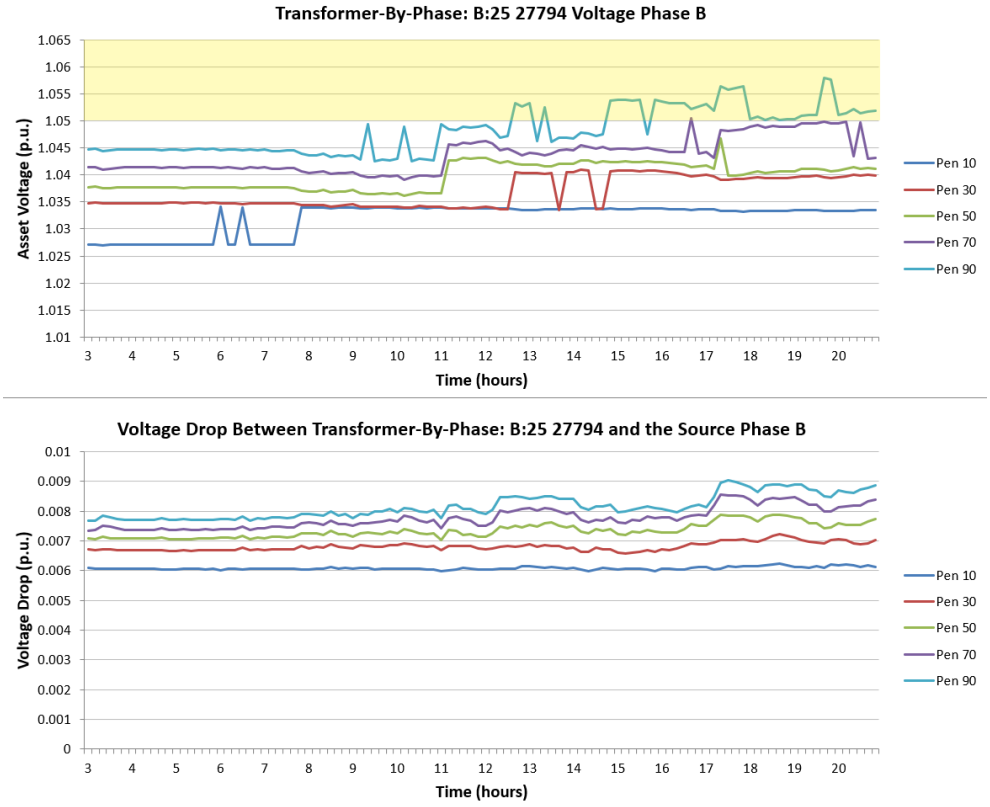

Figure 6.9: Voltage of asset B:25 27794 and delta voltage in respect to the source across EV Penetrations. Voltage drop increases as EV penetration increases.

to asset across each EV penetration. The voltage drop shows the impact of EVSE and electrification even when the input voltage seems incorrect.

6.1.3.2 Single Asset Histogram

As the transformer-by-phase asset A:25 36781 is not often overloaded, its loading histogram is not a good example of tool functionality. The asset A:25 65996 is chosen instead to show the progression of transformer loading events through each EV penetration. This transformer-by-phases histogram is shown in Figure 6.10. Initially, for 10% EV penetration this figure contains no overloaded time periods. Loading events for 30 and 50% EV penetrations last no longer then 2 hours. 70 and 90% EV penetrations however contain loading events in excess of 4 hours. This shows a clear increase in loading event duration and asset danger between the 50 and 70% EV penetrations. Transformer-by-phase loading events increase in duration and frequency as EV penetration increases.

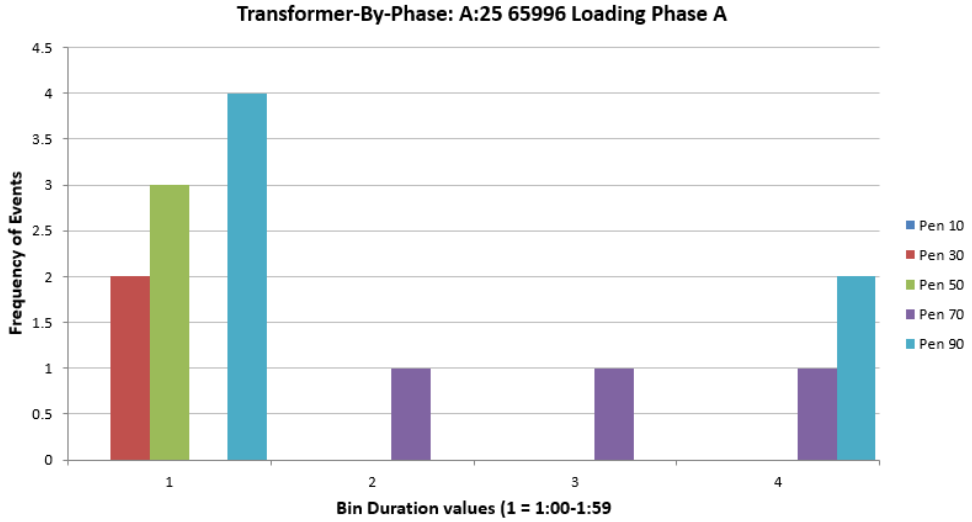

Figure 6.10: Transformer-by-phase loading duration histogram for A:25 65996. Loading event severity increases with EV penetration. Voltage exceeds 1.05 p.u. with 90% EV penetration.

6.1.4 Distribution Asset Voltage and Loading Information

Distribution lines, represented by Overhead-By-Phase lines in CYME, are the second major asset of interest. Distribution lines in the study do not experience a larger amount of overloaded assets. A single asset PRIOH162622-3 was chosen because it feeds the transformer-by-phase A:25 36781, and shows the consistency of voltage from distribution line to transformer. Figure 6.11 shows the loading values for each time step of the simulation. Electrification is the driver of consistent loading increases, causing higher loading values across each time steps due to demand growth. Many noticeable spikes occur in the higher penetration lines. These spikes are due to the addition of EVSE on the customer side of the distribution line. The average loading value for time steps in the evenings are considerably increased with the addition of EVSE loads. This shows distribution lines must be sized to these periods with large EVSE demand, and increases to line loading analyzed for each new

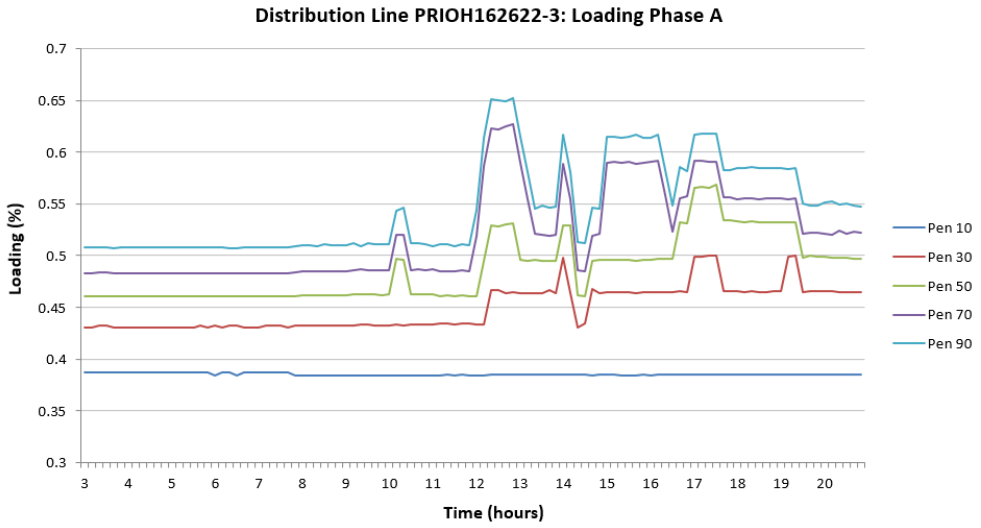

Figure 6.11: Distribution Line loading values for PRIOH162622-3. Loading increases with EV penetration, very low loading.

EVSE planned for household installation. Note the small loading percentage, less then 1%. This shows the line is very oversized, possibly due to incorrect asset properties in CYME.

Voltage levels for this asset are shown in in Figure 6.12. This figure shows decreases in voltage level with increases in EV penetration. Voltage levels vary from 0.967 and 0.931 pu voltage for this distribution line. For EV penetrations 50% and above this asset experiences under-voltage events. At 90% EV penetration this asset experiences an input voltage drop of 1.5% during a day. This figure is very similar to Figure 6.6, as this distribution line feeds only a few different loads besides the transformer-by-phase A:25 36781. Household demand increases due to EVSE results in households make up the large spikes in voltage. Increases in demand cause the input voltage of distribution lines to decrease.

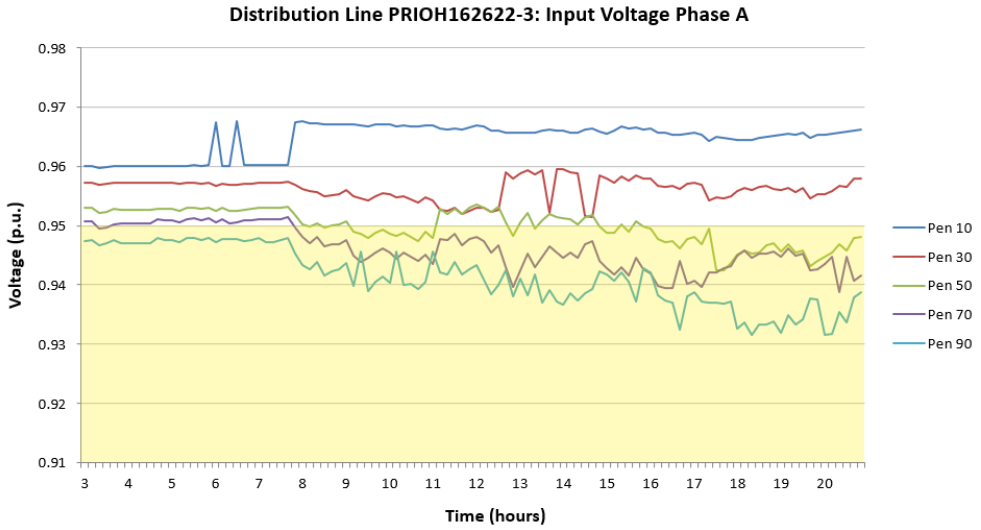

Figure 6.12: Distribution Line voltage values for PRIOH162622-3. Voltage decreases with increased EV penetration. Voltage drops below 0.95 p.u. for high EV penetrations.

6.2 Suite of Tools Capabilities

The case study showcases some of the output data for assets impacted by vehicle electrification. These data can be used for many different purposes. Loading trend data can be used as a rough guideline for how the distribution system transitions in time. This can show when large increases to loading and voltage events may occur.

Each asset has its own CSV output for loading and voltage data. Figures 6.1 to 6.6 were created using the CSV outputs. These CSV include a row for total number of overloaded time steps, and another row for total number of under voltage time steps. These two rows make it simple to sort by interested quality, and allow the impacted assets to be identified. Identifying these problem assets allows designers to conduct studies on equipment to assess if they may be at risk.

To better identify transformer over-loading events, histograms can be examined such as in Figure 6.3. These bins allow the severity of transformer loading events to be understood. This provides extra capability for designers to understand future transformer conditions. These histograms highlight the duration of events, which allow designers to understand which transformers could be at risk of a damaging long duration event.

6.3 Suite of Tools Files

The majority of the files needed to run this suite of tools can be where were they are archived at [38]. This archive includes both the python script used and input CSV's required to run the script. These files are described in A.1. Input CSV's include files like the EV forecast used

Input CSV's	
CarsFromLoad.csv	Compositions relating household demand to number of vehicles
PenetrationVsYears.csv	Forecast EV penetration by year
PEV-Profiles-L1.csv	Demand profiles for Level 1 EVSE with a rating of 6.6 kW
PEV-Profiles-L2.csv	Demand profiles for Level 2 EVSE with a rating of 6.6 kW
Python Code	
definitions.py	File storing different computer address locations used to interface with input CSV's
function_study_analysis.py	File storing functions containing CYME commands in a non-CYME specific value f
LoadFlowOverload.py	File containing functions that record information, process information, and produce CSV outputs
lookup.py	File storing CYME internal asset indexes
main.py	File containing main coding loop, and most functions calls
ModifySpotLoad.py	File containing functions that modify the CYME study
UserInput.py	File containing functions that take user inputs

Table 6.1: Archived tool files [22].

and EVSE demand profiles. Some of the python files contain important information but not functions, such as *definitions.py* and lookup.py. Other python files, like *ModifySpotLoad.py* and *LoadFlowOverload.py* contain functions that drive the capabilities of this suite of tools. These files represent most of the required inputs needed to run this suite of tools. This table and the DOI for the files this suite of tools requires can be found in appendix A.

The CYME study used for this test case, Ceder Hills was a large section of the distribution

network. It included 1248 distribution transformers and over three thousand residential households. As this CYME study, Ceder Hills, is property of PGE it can not be shared. This CYME study is the only required part of this suite of tools that is not provided. While this study is not provided, any valid CYME study can be used with this suite of tools.

7 Conclusion

This work successfully demonstrated the capabilities of a custom suite of tools, along with Python and CYME, to model distribution systems impacted by EVSE load growth. These tools are used to analyze the impacts of EVSE penetration. These impacts may be considered while planning to size EVSE transformers and associated conductors based on current or future loading profiles. Understanding these impacts allows construction of a robust distribution system with effective asset sizes to enable transportation electrification. These tools enable grid operators to anticipate these impacts.

Level 2 EVSE and DCFCs can cause significant reliability issues related to asset overloading and under voltage events. Stochastic renewable generation, which continues to grow, also induce reliability challenges. Demand due to stochastic renewable generation and unexpected EVSE charging may result in distribution line overloads and over current relay trips. This suite of tools may be used by a distribution planner to study the EVSE impact on distribution lines. This, along with a study of nearby stochastic generation, enables planners to properly size distribution lines to remain at or below rated conditions accounting for both stochastic and EVSE line impacts.

Overall this work can apply both residential stochastic EV growth and intentional larger EVSE charging hubs using the *Stochastic Residential EV Tool* and *Intentional EVSE Tool*. In the case of intentional loads, transformers are sized based on EVSE demand. Demand

profile projections are used to simulate EVSE demand through the *Time Series Tool*. The distribution system grows in EVSE and electrification using the *System Growth Tool*. The *Data Collection Tool* provides the output of collected asset information for analysis. These help distribution system analysts to determine the asset conditions and impacts due to an increased EV penetration.

This suite of tool leaves several opportunities for future improvements. One of these opportunities is assessing harmonic impacts on distribution assets. Harmonic impacts may cause a reduced effective rated power, increasing loading values by extension. Another opportunity would be the ability to simulate demand profiles for Level 1 and Level 2 EVSE. This would make it simple to apply different EVSE demand levels otherwise constrained by forecast demand profiles.

A final improvement opportunity is converting the tool from 32 bit to 64 bit versions of Python and CYME. The 32 bit version of Python cannot use more than four GB of RAM. Long simulations both in duration of step sizes and range of EV penetration are impossible to run due to this RAM limitation. Larger time step or EV penetration ranges allow more comprehensive data and analysis, longer time steps for studies, or smaller EV penetration changes providing more data point of EV penetration impacts.

Bibliography

[1] *Transportation Electrification Plan*. Portland General Electric, March 2017.

[2] Office of Transportation and Air Quality. Fast Facts: U.S. Transportation Sector Greenhouse Gas Emissions 1990-2018. June 2020.

[3] M. B. Arias and S. Bae. Electric vehicle charging demand forecasting model based on big data technologies. *Applied Energy*, 183:327–339, September 2016.

[4] I. Babaeiyazdi, A. Rezaei-Zare, and S. Shokrzadeh. Fast charging systems to enable electrification of transportation: An operational constrained based analysis. In *2019 IEEE Transportation Electrification Conference and Expo (ITEC)*, pages 1–6, June 2019.

[5] Z. Tian, T. Jung, Y. Wang, F. Zhang, L. Tu, C. Xu, C. Tian, and X. Li. Real-time charging station recommendation system for electric-vehicle taxis. *IEEE Transactions on Intelligent Transportation Systems*, 17(11):3098–3109, April 2016.

[6] Portland General Electric, corporate standard. *Charging Stations for Electric Vehicles*, June 2018.

[7] Portland General Electric, corporate standard. *Transformers: General Physical and Electrical Characteristics*, February 2019.

[8] Federal Highway Administration. National Household Travel Survey. *U.S. Department of Transportation*, 2017.

[9] Chioke B. Harris and Michael E. Webber. An empirically-validated methodology to simulate electricity demand for electric vehicle charging. *Applied Energy*, 126:172 – 181, 2014.

[10] H. Chin. An analytical evaluation of top-down versus bottom-up forecast in the electricity demand. In *Consumer Electronics-Taiwan (ICCE-TW), 2016 IEEE International Conference on*, May 2016.

[11] F. Therrien, J. S. Lacroix M. Belletête, and M. Reno. Algorithmic Aspects of a Commercial-Grade Distribution System Load Flow Engine. In *2017 IEEE 44th Photovoltaic Specialist Conference (PVSC)*, pages 1579–1584. IEEE, June 2017.

[12] N. Mehnaz, A. I. Bhuiyan, M. Roy, and F. Hossain. Load Flow Analysis And Abnormality Removal Of Bangladesh Power System Using Software CYME PSAF. In *Intelligent Systems, Modelling and Simulation, 4th International Conference on*, volume 1, pages 384–388, July 2013.

[13] A. K. Hamza and M. F. Bonneya. Step Voltage Regulator and Capacitor Placement to Improve the Performance of Rural Electrical Distribution Systems by CYME Program. In *2nd International Conference on Sustainable Engineering Techniques (ICSET 2019)*, volume 518, 2019.

[14] A. Nasiakou, M. Alamaniotis, and L. H. Tsoukalas. MatGridGUI — A toolbox for GridLAB-D simulation platform. In *2016 7th International Conference on Information, Intelligence, Systems Applications (IISA)*, pages 1–5, 2016.

[15] A. Yusuff, N. Ademola, and M. T. Mpumelelo. Enhancement of modeling environment for power distribution network in Gridlab-D on Emacs. In *2019 IEEE AFRICON*, pages 1–6. IEEE, September 2019.

[16] SAE Electric Vehicle and Plug in Hybrid Electric Vehicle Conductive Charge Coupler. October 2017.

[17] H. Tu, H. Feng, S. Srdic, and S. Lukic. Extreme fast charging of electric vehicles: A technology overview. *IEEE Transactions on Transportation Electrification*, 5(4):861–878, December 2019.

[18] What is the Impact of Utility Demand Charges on a DCFC Host? *The EV Project*, June 2015.

[19] A. Kuperman, U. Levy, J. Goren, A. Zafranski, and A. Savernin. High power Li-Ion battery charger for electric vehicle. In *2011 7th International Conference-Workshop Compatibility and Power Electronics (CPE)*, pages 342–347, July 2011.

[20] R. Pawelek, P. Kelm, and I. Wasiak. Experimental Analysis of DC Electric Vehicles Charging Station Operation and its Impact on the Supplying Grid. In *2014 IEEE International Electric Vehicle Conference (IEVC)*, pages 1–4. IEEE, December 2014.

[21] H. Bai, A. Taylor, W. Guo, G. Szatmari-Voicu, N. Wang, J. Patterson, and J. Kane. Design of an 11 kW power factor correction and 10 kW ZVS DC/DC converter for a high-efficiency battery charger in electric vehicles. *IET Power Electronics*, 5(9):1714–1722, November 2012.

[22] K. Clement-Nyns, E. Haesen, and J. Driesen. The Impact of Charging Plug-In Hybrid Electric Vehicles on a Residential Distribution Grid. 25(1):371–380, February 2010.

[23] M. Lillebo, S. Zaferanlouei, A. Zecchio, and H. Farahmand. Impact of large-scale EV integration and fast chargers in a Norwegian LV grid. *The Journal of Engineering*, 2019(18):5104–5108, August 2019.

[24] K. Clement-Nyns, E. Haesen, and J. Driesen. Impact of electric vehicle charging stations on reliability of distribution network. In *2017 International Conference on Technological Advancements in Power and Energy (TAP Energy)*, pages 1–6, December 2017.

[25] N. Woodman, R. B. Bass, and M. Donnelly. Modeling harmonic impacts of electric vehicle chargers on distribution networks. In *2018 IEEE Energy Conversion Congress and Exposition (ECCE)*, pages 2774–2781, September 2018.

[26] A. Malano, S. M. J. Mayer, and S. Bachmann. Harmonic interaction of electric vehicle chargers in a central charging infrastructure. pages 367–372, October 2016.

[27] D.M. Said and K.M. Nor. Effects of harmonics on distribution transformers. In *2008 Australasian Universities Power Engineering Conference*, pages 1–5. IEEE, December 2008.

[28] A. Elmoudi, M. Lehtonen, and H. Nordman. Effect of harmonics on transformers loss of life. In *Conference Record of the 2006 IEEE International Symposium on Electrical Insulation*, pages 408–411, June 2006.

[29] J. C. Gomez and M. M. Morcos. Impact of EV battery chargers on the power quality of distribution systems. *IEEE Transactions on Power Delivery*, 18(3):975–981, July 2003.

[30] S. N. Syed Nasir, J. J. Jamian, and M. W. Mustafa. Minimization of Harmonic Distortion Impact Due to Large-Scale Fast Charging Station Using Modified Lightning Search Algorithm and Pareto-Fuzzy Synergistic Approach. *Electrical And Electronic Engineering, IEEJ Transactions on*, 13(6):815–822, June 2018.

[31] A. Karadeniz and M. E. Balci. Comparative evaluation of various passive filter types regarding their contribution to transformer's loading capacity under non-sinusoidal conditions. In *2017 10th International Conference on Electrical and Electronics Engineering (ELECO)*, pages 1–5, 2017.

[32] T. Thiringer and S. Haghbin. Power Quality Issues of a Battery Fast Charging Station for a Fully-Electric Public Transport System in Gothenburg City. *Batteries*, 1:22–33, November 2015.

[33] S. A. Funke and T. J. Burgert. Electrification Potential of a Taxicab Fleet: A Techno-Economic Case Study from Karlsruhe, Germany. In *2017 IEEE Vehicle Power and Propulsion Conference (VPPC)*, pages 1–7, 2017.

[34] E. Biondi, C. Boldrini, and R. Bruno. The impact of regulated electric fleets on the power grid: The car sharing case. In *2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, pages 1–6, September 2016.

[35] Z. Yi, J. Smart, and M. Shirk. Energy impact evaluation for eco-routing and charging of autonomous electric vehicle fleet: Ambient temperature consideration. *Transportation Research Part C: Emerging Technologies*, 89:344–363, April 2018.

[36] C. Berry. Residential Energy Consumption Survey (RECS). *U.S. Energy Information Administration*, April 2016.

[37] U.S. Census. *Household size by vehicles avaliable*, 2017.

[38] Jacob Sheeran. Data from: Modeling Tools for Analyzing Electrical Power Distribution Systems Impacted by Electric Vehicle Load Growth. 2021. Electrical and Computer Engineering Datasets.2 doi: `https://doi.org/10.15760/ece-data.02`.

Appendix A: Supplementary Files

A.1 DOI

https://doi.org/10.15760/ece-data.02

A.2 Suite of Tools Information

Input CSV's		
CarsFromLoad.csv	Compositions relating household demand to number of vehicles	3 kB
PenetrationVsYears.csv	Forecast EV penetration by year	1 kB
PEV-Profiles-L1.csv	Demand profiles for Level 1 EVSE with a rating of 6.6 kW	44.9 MB
PEV-Profiles-L2.csv	Demand profiles for Level 2 EVSE with a rating of 19.2 kW	39.0 MB
Python Code		
definitions.py	File storing different computer address locations used to interface with input CSV's	1 kB
function_study_analysis.py	File storing functions containing CYME commands in a non-CYME specific value format	2 kB
LoadFlowOverload.py	File containing functions that record information, process information, and produce CSV outputs	114 kB
lookup.py	File storing CYME internal asset indexes	2 kB
main.py	File containing main coding loop, and most functions calls	16 kB
ModifySpotLoad.py	File containing functions that modify the CYME study	38 kB
UserInput.py	File containing functions that take user inputs	8 kB

Table A.1: Archived tool files [22].

A.3 Type of File

A.3.1 Input CSV's

The supplemental files under the Input CSV's catagory all require a file that can read comma separated value files. Excel is a well known program that can read CSV's. A Microsoft 365 subscription is required to download Excel, though a version of it is available on google drive. A free program that one can use to read these files would be Notepad++, which can be downloaded online.

A.3.2 Python Code

Many of the files used for this suite of tools are python coding files. Each of files require a python interpreter to read and compile the script. This suite of tools used the interpreter PyCharm, created by JetBrains, while in development. Another popular interpreter for python and many other coding languages is Anaconda. In my opinion PyCharm is easier to use, but offers less functionality then Anaconda. Both the available programs can be downloaded online, by searching their names.

Appendix B: Python Functions

B.1 HouseholdVehicles

```
def HouseholdVehicles(model_filename, L1_chance, L2_chance):
    '''
    HouseholdCars is designed to assign a certain value related to household load,
        representing multiple cars for multi
    -family households, currently an not arbitraty value
    '''

    path = "C:\\Users\\pwrlab07\\Downloads\\CarsFromLoad.csv"
    df = pandas.read_csv(path)
    OneLoadMember = df['ONE 1']
    TwoLoadMember = df['TWO 1']
    ThreeLoadMember = df['THREE 1']
    FourLoadMember = df['FOUR 1']
    OneMemberCar = df['ONE 2']
    TwoMemberCar = df['TWO 2']
    ThreeMemberCar = df['THREE 2']
    FourMemberCar = df['FOUR 2']


    start2=datetime.now()
    cympy.study.Open(model_filename)

    spotload_pre = function_study_analysis.list_devices(14)

    CustStorage = []
    CustValStorage = []
    OutputStorage=[]
    NameStorage=[]

    #Loops through spotload numbers, each loop name is equal to one of the spotload device
         numbers
    for name in spotload_pre['device_number']:
        CustLoad = cympy.study.GetLoad(name, 14)
        CustList = CustLoad.ListCustomers()
        NewCustAdded=0

        #Loops through each customer index on the specific spotload of name
        for j in range(0, len(CustList)):
            spot_load_device = cympy.study.GetDevice(name, cympy.enums.DeviceType.SpotLoad
                )
            customer_load_prop = "CustomerLoads.Get({value}).".format(value=CustList[j])
            CustType = spot_load_device.GetValue(customer_load_prop + "CustomerType")

            #Looks to see if the customer being accessed is a residental type, which would
                 mean an applicable EV house
```

```
            #Adds a seperate load for storing EV's
            if NewCustAdded ==0:

                #AddCustomerLoad is a broken cympy function, the customer number should be
                     the string input, but is
                #instead the spotload ID
                CustLoad.AddCustomerLoad("blah")
                NewCustAdded = 1

    spotload_pre = function_study_analysis.list_devices(14)

    for name in spotload_pre['device_number']:

        CustLoad = cympy.study.GetLoad(name, 14)
        CustList = CustLoad.ListCustomers()


        for j in range(0, len(CustList)):
            spot_load_device = cympy.study.GetDevice(name, cympy.enums.DeviceType.SpotLoad
                )
            customer_load_prop = "CustomerLoads.Get({value}).".format(value=CustList[j])
            CustType = spot_load_device.GetValue(customer_load_prop + "CustomerType")

            FirstPhrase = "CustomerLoads.Get({num}).CustomerLoadModels.Get(1).".format(num
                =CustList[j])

            #Gathering the phrase for calling on customer loads
            SecondPhrase, kWPhase = PhaseCheck(FirstPhrase, spot_load_device)


            inter_step = [CustList[j]]

            if CustType == "Residential":
                inter_step2 = [float(spot_load_device.GetValue(FirstPhrase + SecondPhrase
                    + "LoadValue.KW"))]

            else:
                inter_step2=[0.0]

            #Each loop appends the found values to the storage variables
            CustStorage.append(inter_step)
            CustValStorage.append(inter_step2)
            NameStorage.append(name)


    OneLoadMember = df['ONE 1']
    TwoLoadMember = df['TWO 1']
    ThreeLoadMember = df['THREE 1']
    FourLoadMember = df['FOUR 1']
    OneMemberCar = df['ONE 2']
    TwoMemberCar = df['TWO 2']
    ThreeMemberCar = df['THREE 2']
    FourMemberCar = df['FOUR 2']
    CarAll=0
    i=0
    ChanceStorage=[]
    for value in CustValStorage:
        value = float(value[0])
```

```
        if value >= 0.0:

            if value < 0.1:
                value = 0
            else:
                value = (UserInput.truncate(value, 1)*10)-1

            value=int(value)
            Pen=random.uniform(0, 100)

            if value <= 74:
                if Pen > 0 and Pen <= OneLoadMember[value]:
                    Members=1

                elif Pen > OneLoadMember[value] and Pen <= (OneLoadMember[value]+
                    TwoLoadMember[value]):
                    Members = 2

                elif Pen > (OneLoadMember[value] + TwoLoadMember[value]) and Pen <= (
                    OneLoadMember[value] + TwoLoadMember[value]+ ThreeLoadMember[value]):
                    Members = 3

                elif Pen > (OneLoadMember[value] + TwoLoadMember[value]+ ThreeLoadMember[
                    value]):
                    Members = 4
            Modify=0
            if value > 74:
                Pen=random.uniform(0, 100)
                if Pen < 15:
                    Members=3
                else:
                    Members=4
                value = 74
                Modify=1
            Pen2 = random.uniform(0, 100)
            Cars=0
            if Members == 1:

                if Pen2 > 0 and Pen2 <= OneMemberCar[0]:
                    Cars=0

                elif Pen2 > OneMemberCar[0] and Pen2 <= (OneMemberCar[0]+OneMemberCar[1]):
                    Cars=1

                elif Pen2 > (OneMemberCar[0] + OneMemberCar[1]) and Pen2 <= (OneMemberCar
                    [0] + OneMemberCar[1]+ OneMemberCar[2]):
                    Cars=2

                elif Pen2 > (OneMemberCar[0] + OneMemberCar[1]+ OneMemberCar[2]) and Pen2
                    <= (OneMemberCar[0] + OneMemberCar[1] + OneMemberCar[2]+ OneMemberCar
                    [3]):
                    Cars = 3
                elif Pen2 > (OneMemberCar[0] + OneMemberCar[1] + OneMemberCar[2]+
                    OneMemberCar[3]):
                    Cars = 4
            if Members == 2:
                if Pen2 > 0 and Pen2 <= TwoMemberCar[0]:
                    Cars=0
```

```
            elif Pen2 > TwoMemberCar[0] and Pen2 <= (TwoMemberCar[0]+TwoMemberCar[1]):
                Cars=1

            elif Pen2 > (TwoMemberCar[0] + TwoMemberCar[1]) and Pen2 <= (TwoMemberCar
                [0] + TwoMemberCar[1]+ TwoMemberCar[2]):
                Cars=2

            elif Pen2 > (TwoMemberCar[0] + TwoMemberCar[1]+ TwoMemberCar[2]) and Pen2
                <= (TwoMemberCar[0] + TwoMemberCar[1] + TwoMemberCar[2]+ TwoMemberCar
                [3]):
                Cars = 3
            elif Pen2 > (TwoMemberCar[0] + TwoMemberCar[1] + TwoMemberCar[2]+
                TwoMemberCar[3]):
                Cars = 4
        if Members == 3:
            if Pen2 > 0 and Pen2 <= ThreeMemberCar[0]:
                Cars=0

            elif Pen2 > ThreeMemberCar[0] and Pen2 <= (ThreeMemberCar[0]+
                ThreeMemberCar[1]):
                Cars=1

            elif Pen2 > (ThreeMemberCar[0] + ThreeMemberCar[1]) and Pen2 <= (
                ThreeMemberCar[0] + ThreeMemberCar[1]+ ThreeMemberCar[2]):
                Cars=2

            elif Pen2 > (ThreeMemberCar[0] + ThreeMemberCar[1]+ ThreeMemberCar[2]) and
                 Pen2 <= (ThreeMemberCar[0] + ThreeMemberCar[1] + ThreeMemberCar[2]+
                ThreeMemberCar[3]):
                Cars = 3
            elif Pen2 > (ThreeMemberCar[0] + ThreeMemberCar[1] + ThreeMemberCar[2]+
                ThreeMemberCar[3]):
                Cars = 4
        if Members == 4:
            if Pen2 > 0 and Pen2 <= FourMemberCar[0]:
                Cars=0

            elif Pen2 > FourMemberCar[0] and Pen2 <= (FourMemberCar[0]+FourMemberCar
                [1]):
                Cars=1

            elif Pen2 > (FourMemberCar[0] + FourMemberCar[1]) and Pen2 <= (
                FourMemberCar[0] + FourMemberCar[1]+ FourMemberCar[2]):
                Cars=2

            elif Pen2 > (FourMemberCar[0] + FourMemberCar[1]+ FourMemberCar[2]) and
                Pen2 <= (FourMemberCar[0] + FourMemberCar[1] + FourMemberCar[2]+
                FourMemberCar[3]):
                Cars = 3
            elif Pen2 > (FourMemberCar[0] + FourMemberCar[1] + FourMemberCar[2]+
                FourMemberCar[3]):
                Cars = 4

        MemberVals=[OneLoadMember[value], OneLoadMember[value] + TwoLoadMember[value],
             OneLoadMember[value] + TwoLoadMember[value] + ThreeLoadMember[value]]

        if Members == 1:
            CarVals=[OneMemberCar[0],OneMemberCar[0] + OneMemberCar[1] ,OneMemberCar
                [0] + OneMemberCar[1] + OneMemberCar[2],OneMemberCar[0] + OneMemberCar
```

```
                    [1] + OneMemberCar[2]+ OneMemberCar[3]]
            elif Members == 2:
                CarVals=[TwoMemberCar[0],TwoMemberCar[0] + TwoMemberCar[1] ,TwoMemberCar
                    [0] + TwoMemberCar[1] + TwoMemberCar[2],TwoMemberCar[0] + TwoMemberCar
                    [1] + TwoMemberCar[2]+ TwoMemberCar[3]]
            elif Members == 3:
                CarVals=[ThreeMemberCar[0],ThreeMemberCar[0] + ThreeMemberCar[1] ,
                    ThreeMemberCar[0] + ThreeMemberCar[1] + ThreeMemberCar[2],
                    ThreeMemberCar[0] + ThreeMemberCar[1] + ThreeMemberCar[2]+
                    ThreeMemberCar[3]]
            elif Members == 4:
                CarVals=[FourMemberCar[0],FourMemberCar[0] + FourMemberCar[1] ,
                    FourMemberCar[0] + FourMemberCar[1] + FourMemberCar[2],FourMemberCar
                    [0] + FourMemberCar[1] + FourMemberCar[2]+ FourMemberCar[3]]

            CarAll=CarAll+Cars
            appendval=[CustStorage[i],value,Pen,Modify,MemberVals,Members,Pen2,CarVals,
                Cars]
            ChanceStorage.append(appendval)

        CarDemandStep= range(0,Cars)
        CarEVNum=[]
        CarEVNumPlaced = []
        CarEVSELevel=[]
        for j in CarDemandStep:
            val = random.uniform(1,348)
            val=int(UserInput.truncate(val,0))
            CarEVNum.append(val)
            rand= random.uniform(0,100)

            if rand >= 0 and rand <= L1_chance:
                Level=1
            elif rand > L1_chance and rand <= (L1_chance + L2_chance):
                Level=2
            CarEVSELevel.append(Level)
        for g in range(len(CarEVNum)):
            Num=0
            CarEVNumPlaced.append(Num)
        OutStore = [CustStorage[i], [Cars,Cars], CarEVNum, CarEVNumPlaced, CarEVSELevel,
            NameStorage[i]]
        OutputStorage.append(OutStore)

        i = i + 1

    #These two lines allow the study name to be changes, allows the base study to remain
        undisturbed
    filename_template = model_filename.split('.')
    filename_changed = filename_template[0] + '_EV_Holding_Created.'+ filename_template[1]

    cympy.study.Save(filename_changed)
    end2 = datetime.now()
    print('HouseHold Cars Done in ' + str((end2 - start2).total_seconds()) + ' seconds')

    return OutputStorage, filename_changed, spotload_pre, CarAll
```

B.2 Add_EV

```
def Add_EV(model_filename,L1,L2,Penetration,through_filename, PEN, CustCarStorage,spotload
    , Type, LaterStorage, L1Store, L2Store, AppliedNames, UnAppliedNames):
    '''
    Currently this only adds a single EV load onto each household, more work could be put
        into use catigorizing the
    houses as different sizes by EV load, and adding the possibility of more EV's on a
        single customer load
    '''
    open_study(model_filename)

    #Calls PenetrationVsYears to apply load growth
    LaterStorage, AppliedNames, UnAppliedNames=PenetrationVsYears(PEN, Type, LaterStorage,
         AppliedNames, UnAppliedNames)
    place=0
    order = -1
    CustOrder=-1
    again=1
    Change=0

    CarsApplied=0



    #arbitrary sized storage variables, will work unless there are more then 3 times as
        many residental customers then
    #there are total spotloads
    LoadValueMatrix = [[]]*len(spotload)*3
    u=0
    NamesApplied=[]
    #Sets LoadValueMa
    for val in LoadValueMatrix:
        LoadValueMatrix[u]=0.0
        u=u+1
    PhraseMaterix = [[]]*len(spotload)*3
    CurrentSpotNum = [[]]*len(spotload)*3
    Placeholder = [[]] * len(spotload) * 3
    CustNumholder = [[]] * len(spotload) * 3

    #While for looping through entire spotload list
    while place < len(spotload):

        spotnum=spotload['device_number'][place]
        CustLoad = cympy.study.GetLoad(spotnum, 14)
        CustList = CustLoad.ListCustomers()
        spot_load_device = cympy.study.GetDevice(spotnum, cympy.enums.DeviceType.SpotLoad)

        if Change == 1:
            order=order+1
            Change = 0

        OriValue=0

        #Loops through each of the customers inside the spotload
        for j in range(0, len(CustList)):
            CustOrder = CustOrder + 1
            ApplyValue=0
            #Check to make sure customer isn't an EV only placement customer
            if CustList[j] != spotnum:
```

```
                customer_load_prop = "CustomerLoads.Get({value}).".format(value=CustList[j
                    ])
                CustType = spot_load_device.GetValue(customer_load_prop + "CustomerType")

                #again = number of EV cars possible
                value=0
                if CustType == "Residential":
                    again = CustCarStorage[CustOrder][1][0]
                    realplace=spotnum
                    PlaceInCust=-1
                    for val in CustCarStorage[CustOrder][3]:
                        PlaceInCust=PlaceInCust+1

                        #Pen is a random value from 0 to 100, checking too see if EV's are
                             added or not
                        Pen=random.uniform(0, 100)
                        if val != 0 and CustCarStorage[CustOrder][4][PlaceInCust] == 1:
                            OriValue = OriValue + L1Store[CustCarStorage[CustOrder][2][
                                PlaceInCust]][0]
                        if val != 0 and CustCarStorage[CustOrder][4][PlaceInCust] == 2:
                            OriValue = OriValue + L2Store[CustCarStorage[CustOrder][2][
                                PlaceInCust]][0]
                        #if the random Pen is larger then the % of added EV's this time,
                            then add the EV load
                        if float(Pen) <= float(Penetration) and val == 0:
                            CarsApplied=CarsApplied+1
                            ApplyValue=ApplyValue+1


                            FirstPhrase = "CustomerLoads.Get({num}).CustomerLoadModels.Get
                                (1).".format(num=realplace)
                            SecondPhrase, kWPhase = PhaseCheck(FirstPhrase,
                                spot_load_device)

                            #Checks too see if val is between 0 and the % ratio of L1 to
                                L2
                            if CustCarStorage[CustOrder][4][PlaceInCust]==1:

                                OriValue =OriValue+L1Store[CustCarStorage[CustOrder][2][
                                    PlaceInCust]][0]/1000

                                CustCarStorage[CustOrder][3][PlaceInCust]=1

                                LoadValueMatrix[order] = OriValue

                                CurrentSpotNum[order] = spotnum
                                PhraseMaterix[order] = FirstPhrase + SecondPhrase + "
                                    LoadValue.KW"
                                Placeholder[order]=place
                                CustNumholder[order]=CustList[j]
                                Change=1


                                #Each time an EV is added, remove it from CustCarStorage
                                CustCarStorage[CustOrder][1][1] = CustCarStorage[CustOrder
                                    ][1][1] - 1
                                value=value+1
```

```
                            #Checks too see if the random val is larger then L1's %, and
                                less then or equal to the
                            #combined L1 and L2 percentages
                            elif CustCarStorage[CustOrder][4][PlaceInCust]==2:

                                OriValue = OriValue + L2Store[CustCarStorage[CustOrder
                                    ][2][PlaceInCust]][0]/1000
                                CustCarStorage[CustOrder][3][PlaceInCust] = 2

                                LoadValueMatrix[order] = OriValue
                                CurrentSpotNum[order] = spotnum
                                PhraseMaterix[order] = FirstPhrase + SecondPhrase + "
                                    LoadValue.KW"
                                Placeholder[order] = place
                                CustNumholder[order] = CustList[j]
                                Change=1



                                CustCarStorage[CustOrder][1][1] = CustCarStorage[CustOrder
                                    ][1][1] - 1
                                value=value+1
            if ApplyValue != 0:
                NamesApplied.append(CustList[j])

        place=place+1

    #ChangeLoad takes the collected customer load manipulation phrases and applies them to
         the EV load
    model_filename_EV = ChangeLoad(LoadValueMatrix, PhraseMaterix, order, CurrentSpotNum,
        through_filename,PEN, Type)

    NamesApplied.sort(key=takeFirst, reverse=True)

    return model_filename_EV, CustCarStorage, CurrentSpotNum, CarsApplied, NamesApplied,
        LaterStorage, AppliedNames, UnAppliedNames
```

B.3 IntentionalLoad

```
def IntentionalLoad(model_filename, IntProfile):
    '''
    This function asks the user if there are Hi-P EVSE they want to add into the study,
        then calls a function for
    creating the branch of equipment required for large EV loads
    '''
    NameStorage=[]

    NamesAll = []
    UnAppliedNames=[]
    AppliedNames=[]
    IntValue =0

    LaterStorage=[]
    start=datetime.now()
    open_study(model_filename)

    #Python User Input
    choice = raw_input("Will you be adding large EVSE? (Yes/No):")
```

```
    while choice == 'Yes':
        #(String)   Case Sensitive Spotload index number
        name = raw_input("Enter Spotload Number/Name:")

        #(int)      Size of EVSE
        demand = raw_input("Enter size of EVSE in kW:")
        demand=float(demand)

        #(int)      Number of same kW EVSE at node to add
        amount = raw_input("Enter number of identically sized EVSE at location:")
        amount=int(amount)
        demandAdjust=demand*amount

        #diversityfactor is from PGE transformer ratings
        diversityfactor = DiversityAdjustment(amount)

        #AdjustedDemand represents total realistic EVSE demand
        AdjustedDemand=demandAdjust*diversityfactor

        #(String)   Phase of EVSE
        phase = raw_input("Enter Phase of EVSE (A,B,C,AB,BC,AC,ABC):")

        #(int)   Phase of EVSE
        volt = int(raw_input("Please enter the secondary voltage of the transformer 208,
            240, 480:"))

        #(string)   If more EVSE of a different size is going to be added
        another=raw_input('Will you be adding any aditional EVSE to this location? (Yes/No
            ):')

        while another == 'Yes':
            #Collects information on the addition EVSE
            demand_add = raw_input("Enter size of EVSE in kW:")
            demand_add = float(demand_add)

            amount_add = raw_input("Enter number of identically sized EVSE at location:")
            amount_add = int(amount_add)
            demandAdjust_add = demand_add * amount_add

            #Finding the diversityfactor for the current class of EVSE, then adds it to
                the tot EVSE demand
            diversityfactor = DiversityAdjustment(amount_add)
            AdjustedDemand_add=demandAdjust_add *diversityfactor
            AdjustedDemand=AdjustedDemand+AdjustedDemand_add

            another = raw_input('Will you be adding any aditional EVSE to this location? (
                Yes/No):')

        if another == 'No':
            #Where intentional EVSE are applied to the distribution study or stored for
                later application
            now=raw_input('Is this load going to be added right away, or wait until a
                certain year?(N/W):')

            if now == 'N':
                # IntLoadBranchCreation takes the information given by the user and
                    creates a seperate
                # branch of transformer and spotload to hold the intentional loads
```

```
                model_filename=IntLoadBranchCreation(name, AdjustedDemand, phase, volt,
                    IntProfile[IntValue])

                #Adds intentional EVSE name to a list of applied intentional EVSE
                value=[name, IntProfile[IntValue]]
                AppliedNames.append(value)

                #List of all EVSE names applied or stored
                NamesAll.append(value)
            else:
                #Asks for the year at which the EVSE will be applied
                year=int(raw_input('What year will this EVSE be added onto the system?:'))

                #Stores the information needed to use IntLoadBranchCreation on a future
                    system
                store= [year, name, AdjustedDemand, phase, volt, 0, IntProfile[IntValue]]
                LaterStorage.append(store)
                value = [name, IntProfile[IntValue]]
                #Adds intentional EVSE name to a list of unapplied intentional EVSE
                UnAppliedNames.append(value)
                NamesAll.append(value)
            IntValue=IntValue+1
        #If yes then asks user for more intentional load information
        choice=raw_input("Will you be adding additional loads at a different spotload? (
            Yes/No):")

    #Creates a new filename, for checking CYME study
    new_filename_template = model_filename.split('.')
    model_filename_changed = new_filename_template[0] + '_Record.' + new_filename_template
        [1]
    save_study(model_filename_changed)
    cympy.study.Close(False)
    end=datetime.now()

    print('IntentionalLoad Done in ' + str((end - start).total_seconds()) + ' seconds')
    return model_filename_changed,  LaterStorage, AppliedNames, UnAppliedNames, NamesAll
```

B.4 IntLoadBranchCreation

```
def IntLoadBranchCreation(Spotload_USER_INPUT, adjustedDemand, phase, volt,DemandProfile):
    '''
    IntLoadBranchCreation takes the user input data and creates the distribution lines and
         transformers connecting the new EVSE load to the distribution system
    '''

    if phase == 'A' or phase == 'B' or phase == 'C':
        TrueDemand = adjustedDemand
    elif phase == 'AB' or phase == 'AC' or phase == 'BC':
        TrueDemand = adjustedDemand/2
    else:
        TrueDemand=adjustedDemand/3

    spotload = cympy.study.ListDevices(14)
    XFMRByPhase = cympy.study.ListDevices(33)
    Cables = cympy.study.ListDevices(10)

    sizelist=[1.5, 3.0, 5.0, 10.0, 15.0, 25.0, 37.5, 50, 55.0, 75.0, 100, 125, 150, 166.7,
         250.0, 333.3, 500.0]
```

```
    for val in sizelist:
        if val > TrueDemand:
            XfmrStartPhrase='{num}_KVA'.format(num=val)
            break

    if volt == 208:
        XfmrEndPhrase='_1P_120/208V'
    elif volt == 240:
        XfmrEndPhrase = '_1P_120/240V'
    elif volt == 480:
        XfmrEndPhrase = '_1P_277/480V'
    else:
        print 'Invalid secondary voltage on load'

    XfmrIDString=XfmrStartPhrase+XfmrEndPhrase

    for abc in range(len(spotload)):
        if spotload[abc].DeviceNumber == Spotload_USER_INPUT:
            spotindex = abc

    CheckingSec = spotload[spotindex].SectionID
    NetworkStr = spotload[spotindex].NetworkID
    load_flow = cympy.sim.LoadFlow()
    load_flow.Run()

    From1 = UserInput.NodeCheckSpot(spotload[spotindex].DeviceNumber)
    #print 'From Spotload Section'
    From2, To2 = UserInput.NodeCheck(spotload, XFMRByPhase, Cables, From1)

    #print 'From Getting XFMR Section'
    #print 'To Getting XFMR Section'
    From3 = UserInput.NodeCheckXFMR(To2)
    #print'From gathering XFMR Device'
    From4, To4 = UserInput.NodeCheck(spotload, XFMRByPhase, Cables, From3)
    #print 'From Getting Cable Section'
    #print 'To Getting Cable Section'
    From5 = UserInput.NodeCheckSection(To4)
    #print'From cable, should be final node'

    XFMRDevice = cympy.study.GetDevice(To2, 33)

    XFMR_IDA = ''
    XFMR_IDB = ''
    XFMR_IDC = ''
    XFMR_ConnectStatus = XFMRDevice.GetValue('ConnectionStatus')
    CableSection = cympy.study.GetDevice(To4, 10)
    Cable_ID = CableSection.GetValue('CableID')
    Cable_length = CableSection.GetValue('Length')

    # Cable Section CableID, Length
    # XFMR section PhaseTransformerID1,2,3, ConnectionStatus

    AnotherStep = cympy.study.GetNode(From5)

    to_node = cympy.study.Node()
    to_node.ID = AnotherStep.ID + '-2'
    to_node.X = AnotherStep.X + 10
    to_node.Y = AnotherStep.Y + 10
    checking = cympy.study.NetworkIterator(From3)
```

```
checking.Next()
Checkingphase = checking.GetSourcePhase()

XFMR_IDA,XFMR_IDB, XFMR_IDC = XfmrID(XFMR_IDA,XFMR_IDB, XFMR_IDC, Checkingphase,
    XfmrIDString)

cympy.study.AddSection(CheckingSec + '-2', NetworkStr, To4 + '-2', cympy.enums.
    DeviceType.Underground,AnotherStep.ID, to_node)

NewCable = cympy.study.GetDevice(To4 + '-2', 10)

NewCable.SetValue(Cable_ID, 'CableID')
NewCable.SetValue(Cable_length, 'Length')

to_node2 = cympy.study.Node()
to_node2.ID = to_node.ID + '-3'

to_node2.X = to_node.X + 10
to_node2.Y = to_node.Y + 10

cympy.study.AddSection(CheckingSec + '-3', NetworkStr, From2 + '-3', cympy.enums.
    DeviceType.TransformerByPhase, to_node.ID,
                        to_node2)

NewXFMR = cympy.study.GetDevice(From2 + '-3', 33)
NewXFMR.SetValue(XFMR_ConnectStatus, 'ConnectionStatus')

if XFMR_IDA != '':
    NewXFMR.SetValue(XFMR_IDA, 'PhaseTransformerID1')
if XFMR_IDB != '':
    NewXFMR.SetValue(XFMR_IDB, 'PhaseTransformerID2')
if XFMR_IDC != '':
    NewXFMR.SetValue(XFMR_IDC, 'PhaseTransformerID3')

load_flow.Run()
to_node3 = cympy.study.Node()
to_node3.ID = to_node2.ID + '-3'
to_node3.X = to_node2.X + 10
to_node3.Y = to_node2.Y + 10

cympy.study.AddSection(CheckingSec + '-4', NetworkStr, To4 + '-4', cympy.enums.
    DeviceType.Underground, to_node2.ID,to_node3)
cympy.study.AddDevice(From2 + '-2', 14, CheckingSec + '-4')

NewLoad= cympy.study.GetLoad(From2+'-2', 14)

NewLoad.AddCustomerLoad('Anything')
NewDevice = cympy.study.GetDevice(From2+'-2', 14)

NewDevice.SetValue('Fixed', 'CustomerLoads.Get({value}).CustomerType'.format(value=
    From2+'-2'))

LoadA="CustomerLoads.Get({num}).CustomerLoadModels.Get(1).CustomerLoadValues.Get(A).
    LoadValue.KW".format(num=From2+'-2')
LoadB="CustomerLoads.Get({num}).CustomerLoadModels.Get(1).CustomerLoadValues.Get(B).
    LoadValue.KW".format(num=From2 + '-2')
LoadC="CustomerLoads.Get({num}).CustomerLoadModels.Get(1).CustomerLoadValues.Get(C).
    LoadValue.KW".format(num=From2 + '-2')
NewDevice.SetValue(float(DemandProfile[0][0]), LoadA)
```

```
    NewDevice.SetValue(float(DemandProfile[1][1]), LoadB)
    NewDevice.SetValue(float(DemandProfile[2][2]), LoadC)


    model_filename_Tests = 'C:\\Users\\pwrlab07\\PycharmProjects\\PGEPython\\INPUT\\
        Ceder_Hills_CheckingAdd.sxst'

    function_study_analysis.save_study(model_filename_Tests)

    return model_filename_Tests
```

B.5 PenetrationVsYears

```
def PenetrationVsYears(Pen, Type, LaterStorage, AppliedNames, UnAppliedNames):
    '''
    PenetrationVsYears reads a csv the lists an estimated EV penetration value next too
        the year when it's expected
    to happen with two columns, one for the year, the other penetration

    This is used for determining the years to apply Load Growth at each penetration
    '''

    UseType='Fixed'
    if Type != '':
        UseType=Type
    path = "C:\\Users\\pwrlab07\\Downloads\\PenetrationVsYear.csv"
    df = pandas.read_csv(path)
    Years_df = df['Year']
    Penetration_df = df['Penetration']
    Pen=int(Pen)

    i=0
    ActualYear=0
    PreviousYear=0
    set = 0
    for val in Penetration_df:
        val=int(val)
        if set == 0:

            if val == Pen:
                ActualYear = int(Years_df[i])
                set=1

            elif (val - Pen) > 0 and i == 0:
                ActualYear = int(Years_df[0])
                set=1

            if i < len(Penetration_df - 1) and set == 0:
                if Pen > Penetration_df[i] and Pen < Penetration_df[i+1]:
                    year_dif=Years_df[i+1]-Years_df[i]
                    pen_dif=Penetration_df[i+1]-Penetration_df[i]
                    val=Pen-Penetration_df[i]
                    ActualYear=val*year_dif/pen_dif + Years_df[i]
                    set=1
        i=i+1

    if LaterStorage != []:
        for i in range(len(LaterStorage)):
```

```
            if LaterStorage[i][0] <= ActualYear and LaterStorage[i][5]==0:

                IntLoadBranchCreation(LaterStorage[i][1],LaterStorage[i][2],LaterStorage[i
                    ][3],LaterStorage[i][4],LaterStorage[i][6])
                appendval=[LaterStorage[i][1], LaterStorage[i][6]]
                AppliedNames.append(appendval)
                integer=-1
                for name in UnAppliedNames:
                    integer = integer+1
                    if name[0] == LaterStorage[i][1]:
                        UnAppliedNames.pop(integer)

                LaterStorage[i][5]=1

    CustTypes = cympy.study.ListCustomerTypes()

    CustTypes.remove(UseType)
    Networks = cympy.study.ListNetworks()
    load_growth = cympy.study.LoadGrowth()
    load_growth.GrowthPercent = 1.5
    load_growth.SetIncludeCustormerType(UseType, False)

    ActualYear=int(math.floor(ActualYear))
    #Check to see if the current year of devices is the same as the year from the imported
        file
    #If it isn't the same year, apply load growth

    if ActualYear != load_growth.GetActualGrowthYears(Networks, CustTypes)[0]:

        load_growth.Apply(Networks, ActualYear)

    load_flow = cympy.sim.LoadFlow()
    load_flow.Run()

    return LaterStorage,AppliedNames, UnAppliedNames
```

B.6 Reapply

```
def ReApply(model_filename, CustCarStorage, time, L1Store, L2Store, AppliedIntNames,
    time_int):
    '''
    This function exists solely to reapplying different loads at different time steps
    '''
    load_flow = cympy.sim.LoadFlow()
    load_flow.Run()
    NamesApplied=[]
    place=0
    TotalApplied=0
    secondstep = 0
    for cust in CustCarStorage:

        OriValue=0
        if secondstep==1:
            if cust[5] == OldCust:
                OriValue=LastAdd
        valuePlace=-1
        CheckPlaced=-1
        for value in cust[3]:
```

```
            valuePlace=valuePlace+1
            if value==1:
                CheckPlaced=1
                OriValue=OriValue+L1Store[cust[2][valuePlace]][time]/1000
            if value==2:
                CheckPlaced = 2
                OriValue=OriValue+L2Store[cust[2][valuePlace]][time]/1000
            LastAdd=OriValue
            OldCust=cust[5]
            secondstep=1


        spot_load_device = cympy.study.GetDevice(cust[5], cympy.enums.DeviceType.SpotLoad)
        place=place+1
        if CheckPlaced != -1:
            NamesApplied.append(cust[0][0])
        FirstPhrase = "CustomerLoads.Get({num}).CustomerLoadModels.Get(1).".format(num=
            cust[5])
        SecondPhrase, kWPhase = PhaseCheck(FirstPhrase, spot_load_device)
        phrase=FirstPhrase + SecondPhrase + "LoadValue.KW"

        spot_load_device.SetValue(float(OriValue), phrase)
        TotalApplied=TotalApplied+int(OriValue)

    time_int
    for value in AppliedIntNames:
        intspotload=value[0]+'-2'
        spot_load_device = cympy.study.GetDevice(intspotload, cympy.enums.DeviceType.
            SpotLoad)
        FirstPhrase = "CustomerLoads.Get({num}).CustomerLoadModels.Get(1).".format(num=
            cust[5])

        LoadA = "CustomerLoads.Get({num}).CustomerLoadModels.Get(1).CustomerLoadValues.Get
            (A).LoadValue.KW".format(num=intspotload)
        LoadB = "CustomerLoads.Get({num}).CustomerLoadModels.Get(1).CustomerLoadValues.Get
            (B).LoadValue.KW".format(num=intspotload)
        LoadC = "CustomerLoads.Get({num}).CustomerLoadModels.Get(1).CustomerLoadValues.Get
            (C).LoadValue.KW".format(num=intspotload)

        spot_load_device.SetValue(float(value[1][0][time_int]), LoadA)
        spot_load_device.SetValue(float(value[1][1][time_int]), LoadB)
        spot_load_device.SetValue(float(value[1][2][time_int]), LoadC)

    NamesApplied.sort(key=takeFirst, reverse=True)
    return NamesApplied
```

B.7 Processing

```
def Processing(xfmrStorage, xfmr_byphaseStorage, cableStorage, TimeRange, XfmrVolt,
    ByPhaseVolt, CableVolt, IntByPhaseStorage,IntByPhaseVolt, NamesAll, HowMany,
    ByPhaseVoltDrop, CableVoltDrop, XfmrVoltOut):

    '''
    Processing works takes the raw floating point number data and converts it into a form
        that can be used to output CSV's
    '''

    XfmrStorageFull, XfmrLenOverFull, XfmrVoltStorageFull, XfmrVoltOutput,XfmrVoltageTrend
        , XfmrLoadingTrend, WorstValues=OverloadGathering(xfmrStorage, TimeRange, XfmrVolt
        )

    XfmrMostStore=HowManyStoring(WorstValues)

    XfmrExcel,ExcelNameHolder, XfmrExcelHM=ExcelFormatCreation(xfmrStorage,XfmrMostStore,
        XfmrStorageFull,XfmrVoltStorageFull, HowMany)

    XfmrVoltExcel,ExcelNameHolderVolt,XfmrVoltExcelHM=ExcelFormatCreation(XfmrVolt,
        XfmrMostStore,XfmrStorageFull,XfmrVoltStorageFull, HowMany)

    XfmrVoltOutExcel,Nul1,Nul2=ExcelFormatCreation(XfmrVoltOut,XfmrMostStore,
        XfmrStorageFull,XfmrVoltStorageFull, HowMany)

    XfmrHist,XfmrGraph,XfmrHistHM=HistogramFormat(XfmrLenOverFull, XfmrMostStore, HowMany)

    ByPhaseStorageFull, ByPhaseLenOverFull,ByPhaseVoltStorageFull, ByPhaseVoltOutput,
        ByPhaseVoltageTrend, ByPhaseLoadingTrend, WorstValues=OverloadGathering(
        xfmr_byphaseStorage, TimeRange, ByPhaseVolt)

    Dropvar1, Null1, Dropvar2, Null2,Null3, Null4, Null5=OverloadGathering(
        xfmr_byphaseStorage, TimeRange, ByPhaseVoltDrop)

    ByPhaseMostStore=HowManyStoring(WorstValues)

    ByPhaseExcel,ExcelNameHolder, ByPhaseExcelHM = ExcelFormatCreation(xfmr_byphaseStorage
        , ByPhaseMostStore, ByPhaseStorageFull,ByPhaseVoltStorageFull, HowMany)

    ByPhaseVoltExcel,ExcelNameHolder, ByPhaseVoltExcelHM = ExcelFormatCreation(ByPhaseVolt
        , ByPhaseMostStore, ByPhaseStorageFull,ByPhaseVoltStorageFull, HowMany)

    ByPhaseVoltDropExcel, Null1, Null2 = ExcelFormatCreation(ByPhaseVoltDrop,
        ByPhaseMostStore, Dropvar1, Dropvar2, HowMany)

    ByPhaseHist,ByPhaseGraph,ByPhaseHistHM = HistogramFormat(ByPhaseLenOverFull,
        ByPhaseMostStore, HowMany)

    if NamesAll != []:
        IntByPhaseStorageFull, IntByPhaseLenOverFull,IntByPhaseVoltStorageFull,
            IntByPhaseVoltOutput,IntByPhaseVoltageTrend, IntByPhaseLoadingTrend,
            WorstValues=OverloadGathering(IntByPhaseStorage, TimeRange, IntByPhaseVolt)

        IntByPhaseMostStore=HowManyStoring(WorstValues)

        IntByPhaseExcel,ExcelNameHolder ,IntByPhaseExcelHM= ExcelFormatCreation(
            IntByPhaseStorage, IntByPhaseMostStore, IntByPhaseStorageFull,
```

```
        IntByPhaseVoltStorageFull, HowMany)

    IntByPhaseVoltExcel,ExcelNameHolder,IntByPhaseVoltExcelHM = ExcelFormatCreation(
        IntByPhaseVolt, IntByPhaseMostStore, IntByPhaseStorageFull,
        IntByPhaseVoltStorageFull, HowMany)

    IntByPhaseHist,IntByPhaseGraph,IntByPhaseHistHM = HistogramFormat(
        IntByPhaseLenOverFull, IntByPhaseMostStore, HowMany)
else:
    IntByPhaseExcel=0
    IntByPhaseMostStore=0
    IntByPhaseVoltExcel=0
    IntByPhaseHist=0
    IntByPhaseExcelHM=0
    IntByPhaseVoltExcelHM=0
    IntByPhaseHistHM=0

CableStorageFull, CableLenOverFull,CableVoltStorageFull, CableVoltOutput,
    CableVoltageTrend, CableLoadingTrend, WorstValues=OverloadGathering(cableStorage,
    TimeRange, CableVolt)

CableMostStore=HowManyStoring(WorstValues)

CableExcel,ExcelNameHolder,CableExcelHM = ExcelFormatCreation(cableStorage,
    CableMostStore, CableStorageFull,CableVoltStorageFull, HowMany)

CableVoltExcel,ExcelNameHolder, CableVoltExcelHM= ExcelFormatCreation(CableVolt,
    CableMostStore, CableStorageFull, CableVoltStorageFull, HowMany)
CableVoltDropExcel=[]

CableHist, CableGraph,CableHistHM = HistogramFormat(CableLenOverFull,CableMostStore,
    HowMany)


FullVoltage=[]
FullVoltage.append(XfmrVoltOutput)
FullVoltage.append(ByPhaseVoltOutput)
FullVoltage.append(CableVoltOutput)
if NamesAll != []:
    FullVoltage.append(IntByPhaseVoltOutput)
GraphStorage=[]
GraphStorage.append(XfmrGraph)
GraphStorage.append(ByPhaseGraph)
GraphStorage.append(CableGraph)
if NamesAll != []:
    GraphStorage.append(IntByPhaseGraph)
VoltageTrendStore=[]
VoltageTrendStore.append(XfmrVoltageTrend)
VoltageTrendStore.append(ByPhaseVoltageTrend)
VoltageTrendStore.append(CableVoltageTrend)
if NamesAll != []:
    VoltageTrendStore.append(IntByPhaseVoltageTrend)
LoadingTrendStore=[]
LoadingTrendStore.append(XfmrLoadingTrend)
LoadingTrendStore.append(ByPhaseLoadingTrend)
LoadingTrendStore.append(CableLoadingTrend)
if NamesAll != []:
    LoadingTrendStore.append(IntByPhaseLoadingTrend)
```

```
    xfmrStorage=0
    xfmr_byphaseStorage=0
    cableStorage=0
    XfmrVolt=0
    ByPhaseVolt=0
    CableVolt=0
    IntByPhaseStorage=0
    IntByPhaseVolt=0

    XfmrVoltOutput = 0
    ByPhaseVoltOutput = 0
    CableVoltOutput = 0
    IntByPhaseVoltOutput = 0

    XfmrGraph = 0
    ByPhaseGraph = 0
    CableGraph = 0
    IntByPhaseGraph = 0

    XfmrVoltageTrend = 0
    ByPhaseVoltageTrend = 0
    CableVoltageTrend = 0
    IntByPhaseVoltageTrend = 0

    XfmrLoadingTrend = 0
    ByPhaseLoadingTrend = 0
    CableLoadingTrend = 0
    IntByPhaseLoadingTrend = 0
    XfmrStorageFull=0
    XfmrLenOverFull=0
    XfmrVoltStorageFull=0
    ByPhaseStorageFull = 0
    ByPhaseLenOverFull = 0
    ByPhaseVoltStorageFull = 0
    CableStorageFull = 0
    CableLenOverFull = 0
    CableVoltStorageFull = 0
    IntByPhaseStorageFull = 0
    IntByPhaseLenOverFull = 0
    IntByPhaseVoltStorageFull = 0
    ExcelNameHolder=0
    CableVoltDrop=0
    ByPhaseVoltDrop=0

    return XfmrExcel, XfmrMostStore,XfmrVoltExcel, ByPhaseExcel, ByPhaseMostStore,
        ByPhaseVoltExcel,CableExcel, CableMostStore,CableVoltExcel, XfmrHist,ByPhaseHist,
        FullVoltage,GraphStorage,IntByPhaseExcel, IntByPhaseMostStore,IntByPhaseVoltExcel,
        IntByPhaseHist, VoltageTrendStore, LoadingTrendStore,XfmrExcelHM,XfmrVoltExcelHM,
        XfmrHistHM,ByPhaseExcelHM,ByPhaseVoltExcelHM,ByPhaseHistHM,IntByPhaseExcelHM,
        IntByPhaseVoltExcelHM,IntByPhaseHistHM,CableExcelHM,CableVoltExcelHM,CableHistHM,
        ByPhaseVoltDropExcel, CableVoltDropExcel , XfmrVoltOutExcel
```

B.8 CSVOutputs

```
def CSVOutputs(Penetration, XfmrExcel,XfmrWorst, ByPhaseExcel,ByPhaseWorst, CableExcel,
    CableWorst, XfmrVoltExcel, ByPhaseVoltExcel, CableVoltExcel,XfmrHist,ByPhaseHist,
    FullVoltage,GraphStorage,IntByPhaseExcel,IntByPhaseWorst, IntByPhaseVoltExcel,
    IntByPhaseHist, VoltageTrendStore, LoadingTrendStore, NamesAll, TimeRange,XfmrExcelHM,
    XfmrVoltExcelHM,XfmrHistHM,ByPhaseExcelHM,ByPhaseVoltExcelHM,ByPhaseHistHM,
    IntByPhaseExcelHM,IntByPhaseVoltExcelHM,IntByPhaseHistHM,CableExcelHM,CableVoltExcelHM
    ,CableHistHM,times, ByPhaseVoltDropExcel,CableVoltDropExcel,XfmrVoltOutExcel):

    '''
    CSVOutputs is the function used to output information found during the study into CSV'
        s saved to the users drive
    '''

    timesExcel=times
    timesExcel.insert(0, '1/1/2010 0:00')

    timesExcel.insert(0, '1/1/2010 0:00')

    name = ('Transformer_Loading.xlsx')
    ExportExcel(Penetration,timesExcel, XfmrExcel,XfmrWorst, name)
    XfmrExcel=0

    name = ('By_Phase_Transformer_Loading.xlsx')
    ExportExcel(Penetration,timesExcel, ByPhaseExcel,ByPhaseWorst, name)
    ByPhaseExcel = 0

    name = ('Transmission_Line_Loading.xlsx')
    ExportExcel(Penetration,timesExcel, CableExcel,CableWorst, name)
    CableExcel = 0

    name = ('Transformer_Voltage_Level.xlsx')
    ExportExcel(Penetration,timesExcel, XfmrVoltExcel,XfmrWorst, name)
    XfmrVoltExcel=0
    name = ('Transformer_Voltage_Level_Distribution_Side.xlsx')
    ExportExcel(Penetration,timesExcel, XfmrVoltOutExcel,XfmrWorst, name)
    XfmrVoltExcel=0

    name = ('By_Phase_Transformer_Voltage_Level.xlsx')
    ExportExcel(Penetration,timesExcel, ByPhaseVoltExcel,ByPhaseWorst, name)
    ByPhaseVoltExcel=0

    name = ('By_Phase_Transformer_Voltage_Drop.xlsx')
    ExportExcel(Penetration,timesExcel, ByPhaseVoltDropExcel,ByPhaseWorst, name)
    ByPhaseVoltDropExcel=0

    name = ('Transmission_Line_Voltage_Level.xlsx')
    ExportExcel(Penetration,timesExcel, CableVoltExcel,CableWorst, name)
    CableVoltExcel=0

    if NamesAll != []:
        name = ('Intentional_By_Phase_Transformer_Loading.xlsx')
        ExportExcel(Penetration,timesExcel, IntByPhaseExcel,IntByPhaseWorst, name)
        IntByPhaseExcel=0
        name = ('Intentional_By_Phase_Transformer_Loading_Worst_Assets.xlsx')
        ExportExcel(Penetration, timesExcel, IntByPhaseExcelHM, IntByPhaseWorst, name)
        IntByPhaseExcelHM = 0
```

```
        name = ('Intentional_By_Phase_Transformer_Voltage_Level.xlsx')
        ExportExcel(Penetration,timesExcel, IntByPhaseVoltExcel,IntByPhaseWorst, name)
        IntByPhaseVoltExcel=0
        name = ('Intentional_By_Phase_Transformer_Voltage_Level_Worst_Assets.xlsx')
        ExportExcel(Penetration,timesExcel, IntByPhaseVoltExcelHM,IntByPhaseWorst, name)
        IntByPhaseVoltExcelHM=0

    val = -1
    for Pen in Penetration:
        val = val + 1
        excelTitle = 'Transformer_Loading_Histograms_{penetration}.xlsx'.format(
            penetration=Pen)
        ExportHistogram(XfmrHist,excelTitle,val)
    XfmrHist=0
    val = -1
    for Pen in Penetration:
        val = val + 1
        excelTitleByPhase = 'By_Phase_Transformer_Loading_Histograms_{penetration}.xlsx'.
            format(penetration=Pen)
        ExportHistogram(ByPhaseHist, excelTitleByPhase, val)
    ByPhaseHist=0
    val = -1
    for Pen in Penetration:
        val = val + 1
        excelTitle = 'Transformer_Loading_Histograms_{penetration}_Worst_Assets.xlsx'.
            format(penetration=Pen)
        ExportHistogram(XfmrHistHM,excelTitle,val)
    XfmrHistHM=0
    val = -1
    for Pen in Penetration:
        val = val + 1
        excelTitleByPhase = 'By_Phase_Transformer_Loading_Histograms_{penetration}
            _Worst_Assets.xlsx'.format(penetration=Pen)
        ExportHistogram(ByPhaseHistHM, excelTitleByPhase, val)
    ByPhaseHistHM=0
    if NamesAll != []:
        val = -1
        for Pen in Penetration:
            val = val + 1
            excelTitleByPhase = 'Intentional_By_Phase_Transformer_Loading_Histograms_{
                penetration}.xlsx'.format(penetration=Pen)
            ExportHistogram(IntByPhaseHist, excelTitleByPhase, val)
        IntByPhaseHist=0
        val = -1
        for Pen in Penetration:
            val = val + 1
            excelTitleByPhase = 'Intentional_By_Phase_Transformer_Loading_Histograms_{
                penetration}_Worst_Assets.xlsx'.format(
                penetration=Pen)
            ExportHistogram(IntByPhaseHistHM, excelTitleByPhase, val)
        IntByPhaseHistHM = 0

    excelTitle = 'Asset_Under_Voltage_Trends_Use_Case.xlsx'
    VoltageOutput(FullVoltage, Penetration,excelTitle)
    FullVoltage=0
    excelTitle = 'Asset_Under_Voltage_Trends_Use_Case_Per_Pen.xlsx'
    VoltageOutputPen(VoltageTrendStore, Penetration, excelTitle, times)
    excelTitle = 'Asset_Over_Loading_Trends_Use_Case_Per_Pen.xlsx'
    LoadingOutputPen(LoadingTrendStore, Penetration, excelTitle, times)
```

```
    VoltageTrendStore=0

    LoadingGraphOutput(GraphStorage, Penetration)

    return
```